AdresseManager

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.adresse;

import java.util.List;

import ch.hsluw.mangelmanager.model.Adresse;


/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface AdresseManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Adresse add(Adresse entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Adresse update(Adresse entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Adresse entity) throws Exception;

    /**
     * Löscht die Entity mit der übegebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Adresse findById(Integer id);
```

```
    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Adresse> findAll();
}
```

AdresseManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.adresse;
import java.util.List;

import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.persister.dao.adresse.AdresseDAO;
import ch.hsluw.mangelmanager.persister.dao.adresse.AdresseDAOImpl;




/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * AdresseManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class AdresseManagerImpl implements AdresseManager {

    private AdresseDAO adresseDAO = new AdresseDAOImpl();

    @Override
    public Adresse add(Adresse entity) throws Exception {

        if (entity.getId() == null) {
            adresseDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Adresse update(Adresse entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return adresseDAO.update(entity);
        }
    }
```

```
    @Override
    public void delete(Adresse entity) throws Exception {
        adresseDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub
        adresseDAO.deleteAdresseById(id);
    }

    @Override
    public Adresse findById(Integer id) {
        return adresseDAO.findAdresseById(id);
    }

    @Override
    public List<Adresse> findAll() {
        return adresseDAO.findAllAdresse();
    }



}
```

ArbeitstypManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.arbeitstyp;

import java.util.List;

import ch.hsluw.mangelmanager.model.Arbeitstyp;

/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ArbeitstypManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Arbeitstyp add(Arbeitstyp entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
```

```
    Arbeitstyp update(Arbeitstyp entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Arbeitstyp entity) throws Exception;

    /**
     * Löscht die Entity mit der übegebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Arbeitstyp findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Arbeitstyp> findAll();

    /**
     * Liefert die Liste mit Arbeitstypen für die übergebene Bezeichnung
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    List<Arbeitstyp> findByBezeichnung(String bezeichnung);

}
```

ArbeitstypManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.arbeitstyp;
import java.util.List;

import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.persister.dao.arbeitstyp.ArbeitstypDAO;
import ch.hsluw.mangelmanager.persister.dao.arbeitstyp.ArbeitstypDAOImpl;
```

```java
/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ArbeitstypManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ArbeitstypManagerImpl implements ArbeitstypManager {

    private ArbeitstypDAO arbeitstypDAO = new ArbeitstypDAOImpl();

    @Override
    public Arbeitstyp add(Arbeitstyp entity) throws Exception {

        if (entity.getId() == null) {
            arbeitstypDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Arbeitstyp update(Arbeitstyp entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return arbeitstypDAO.update(entity);
        }
    }

    @Override
    public void delete(Arbeitstyp entity) throws Exception {
        arbeitstypDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub
        arbeitstypDAO.deleteArbeitstypById(id);
    }

    @Override
    public Arbeitstyp findById(Integer id) {
        return arbeitstypDAO.findArbeitstypById(id);
    }

    @Override
    public List<Arbeitstyp> findAll() {
        return arbeitstypDAO.findAllArbeitstyp();
    }

    @Override
    public List<Arbeitstyp> findByBezeichnung(String bezeichnung) {
        return arbeitstypDAO.findArbeitstypByBezeichnung(bezeichnung);
    }
```

```
}
```

BauherrManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.bauherr;

import java.util.List;

import ch.hsluw.mangelmanager.model.Bauherr;


/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface BauherrManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Bauherr add(Bauherr entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Bauherr update(Bauherr entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Bauherr entity) throws Exception;

    /**
     * Löscht die Entity mit der übegebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
```

```
     * @return
     */
    Bauherr findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Bauherr> findAll();
}
```

BauherrManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.bauherr;
import java.util.List;

import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.persister.dao.bauherr.BauherrDAO;
import ch.hsluw.mangelmanager.persister.dao.bauherr.BauherrDAOImpl;




/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * BauherrManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class BauherrManagerImpl implements BauherrManager {

    private BauherrDAO adresseDAO = new BauherrDAOImpl();

    @Override
    public Bauherr add(Bauherr entity) throws Exception {

        if (entity.getId() == null) {
            adresseDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Bauherr update(Bauherr entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
```

```
            return adresseDAO.update(entity);
        }
    }

    @Override
    public void delete(Bauherr entity) throws Exception {
        adresseDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub
        adresseDAO.deleteBauherrById(id);
    }

    @Override
    public Bauherr findById(Integer id) {
        return adresseDAO.findBauherrById(id);
    }

    @Override
    public List<Bauherr> findAll() {
        return adresseDAO.findAllBauherr();
    }



}
```

GuMitarbeiterManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.gumitarbeiter;

import java.util.List;

import ch.hsluw.mangelmanager.model.GuMitarbeiter;

/**
 * Interface fuer GuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface GuMitarbeiterManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    GuMitarbeiter add(GuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
```

```
     * @param entity
     * @return
     * @throws Exception
     */
    GuMitarbeiter update(GuMitarbeiter entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(GuMitarbeiter entity) throws Exception;

    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    GuMitarbeiter findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<GuMitarbeiter> findAll();
}
```

GuMitarbeiterManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.gumitarbeiter;
import java.util.List;

import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.persister.dao.gumitarbeiter.GuMitarbeiterDAO;
import
ch.hsluw.mangelmanager.persister.dao.gumitarbeiter.GuMitarbeiterDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * GuMitarbeiterManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class GuMitarbeiterManagerImpl implements GuMitarbeiterManager {

    private GuMitarbeiterDAO guMitarbeiterDAO = new GuMitarbeiterDAOImpl();

    @Override
    public GuMitarbeiter add(GuMitarbeiter entity) throws Exception {
```

```
        if (entity.getId() == null) {
            guMitarbeiterDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public GuMitarbeiter update(GuMitarbeiter entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return guMitarbeiterDAO.update(entity);
        }
    }

    @Override
    public void delete(GuMitarbeiter entity) throws Exception {
        guMitarbeiterDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        guMitarbeiterDAO.deleteGuMitarbeiterById(id);

    }

    @Override
    public GuMitarbeiter findById(Integer id) {
        return guMitarbeiterDAO.findGuMitarbeiterById(id);
    }

    @Override
    public List<GuMitarbeiter> findAll() {
        return guMitarbeiterDAO.findAllGuMitarbeiter();
    }
}
```

LoginManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.login;

import java.util.List;

import ch.hsluw.mangelmanager.model.Login;

/**
 * Interface fuer Login Entity
 *
 * @version 1.0
 * @author miten
 *
```

```
 */
public interface LoginManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Login add(Login entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Login update(Login entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Login entity) throws Exception;

    /**
     * Löscht die Entity mit übergebener Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Login findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Login> findAll();

    Login findByName(String name);
}
```

LoginManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
package ch.hsluw.mangelmanager.business.login;
import java.util.List;

import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.persister.dao.login.LoginDAO;
import ch.hsluw.mangelmanager.persister.dao.login.LoginDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * LoginManager zur Verfügung.
 *
 * @version 1.0
 * @author miten
 *
 */
public class LoginManagerImpl implements LoginManager {

    private LoginDAO loginDAO = new LoginDAOImpl();

    @Override
    public Login add(Login entity) throws Exception {

        if (entity.getId() == null) {
            loginDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Login update(Login entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return loginDAO.update(entity);
        }
    }

    @Override
    public void delete(Login entity) throws Exception {
        loginDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        loginDAO.deleteLoginById(id);

    }

    @Override
    public Login findById(Integer id) {
        return loginDAO.findLoginById(id);
    }

    @Override
    public List<Login> findAll() {
        return loginDAO.findAllLogin();
```

```
    }

    @Override
    public Login findByName(String name) {
        // TODO Auto-generated method stub
        return loginDAO.findByName(name);
    }
}
```

MangelManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.mangel;

import java.util.Date;
import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;

/**
 * Interface fuer Mangel Entity
 *
 * @version 1.0
 * @author mmont
 *
 */
public interface MangelManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Mangel add(Mangel entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Mangel update(Mangel entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Mangel entity) throws Exception;

    /**
     * Löscht die Entity mit übergebener Id.
     *
     * @param id
     * @throws Exception
```

```
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Mangel findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Mangel> findAll();

    /**
     * Liefert die Liste mit Mängeln für die übergebene Bezeichnung zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    List<Mangel> findByBezeichnung(String bezeichnung);

    /**
     * Liefert die Liste mit Mängeln für den übergebenen Mangelstatus
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param mangelstatus
     * @return
     */
    List<Mangel> findByMangelstatus(String mangelstatus);

    /**
     * Liefert die Liste mit Mängeln für den übergebenen Erfassungszeit
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param erfassungsZeit
     * @return
     */
    List<Mangel> findByErfassungsZeit(Date erfassungsZeit);

    /**
     * Liefert die Liste mit Mängeln für die übergebene Faelligkeitsdatum
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param faelligkeitsDatum
     * @return
     */
    List<Mangel> findByFaelligkeitsDatum(Date faelligkeitsDatum);

    /**
     * Liefert die Liste mit Mängeln für den übergebenen Abschlusszeit
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param AbschlussZeit
```

```
     * @return
     */
    List<Mangel> findByAbschlussZeit(Date abschlussZeit);

    /** Liefer die Liste mit Mängeln für den übergebenen Namen zurück,
     * falls welche gefunden, sonst eine leere Liste.
     * @param name
     * @return
     */
    List<Mangel> findByName(String name);

    /**
     * Liefert alle Mängel vom Projekt
     * @param projekt
     * @return
     */
    List<Mangel> findAllMangelProjekt(Integer projekt);

}
```

MangelManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.mangel;

import java.util.Date;
import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.persister.dao.mangel.MangelDAO;
import ch.hsluw.mangelmanager.persister.dao.mangel.MangelDAOImpl;

/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MangelManager zur Verfügung.
 *
 * @version 1.0
 * @author mmont
 *
 */
public class MangelManagerImpl implements MangelManager {

    private MangelDAO mangelDAO = new MangelDAOImpl();

    @Override
    public Mangel add(Mangel entity) throws Exception {

        if (entity.getId() == null) {
            mangelDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
```

```
public Mangel update(Mangel entity) throws Exception {

    if (entity.getId() == null) {
        return add(entity);
    } else {
        return mangelDAO.update(entity);
    }
}

@Override
public void delete(Mangel entity) throws Exception {
    mangelDAO.delete(entity);
}

@Override
public void deleteById(Integer id) throws Exception {
    mangelDAO.deleteMangelById(id);

}

@Override
public Mangel findById(Integer id) {
    return mangelDAO.findMangelById(id);
}

@Override
public List<Mangel> findAll() {
    return mangelDAO.findAllMangel();
}

@Override
public List<Mangel> findByBezeichnung(String bezeichnung) {
    return mangelDAO.findMangelByBezeichnung(bezeichnung);
}

@Override
public List<Mangel> findByMangelstatus(String mangelstatus) {
    return mangelDAO.findMangelByMangelstatus(mangelstatus);
}

@Override
public List<Mangel> findByErfassungsZeit(Date erfassungsZeit) {
    return mangelDAO.findMangelByErfassungszeit(erfassungsZeit);
}

@Override
public List<Mangel> findByFaelligkeitsDatum(Date faelligkeitsDatum) {
    return mangelDAO.findMangelByFaelligkeitsDatum(faelligkeitsDatum);
}

@Override
public List<Mangel> findByAbschlussZeit(Date abschlussZeit) {
    return mangelDAO.findMangelByAbschlussZeit(abschlussZeit);
}

@Override
public List<Mangel> findByName(String name) {
    return mangelDAO.findMangelByName(name);
}

@Override
public List<Mangel> findAllMangelProjekt(Integer projekt) {
    return mangelDAO.findAllMangelProjekt(projekt);
```

```
    }

}
```

MangelstatusManager

```
/*
 * ZWECK: Mangelstatusmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.mangelstatus;

import java.util.List;

import ch.hsluw.mangelmanager.model.Mangelstatus;


/**
 * Interface fuer Mangelstatus Entity
 *
 * @version 1.0
 * @author mmont
 *
 */
public interface MangelstatusManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Mangelstatus add(Mangelstatus entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Mangelstatus update(Mangelstatus entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Mangelstatus entity) throws Exception;

    List<Mangelstatus> getAllMangelstatus() throws Exception;
}
```

MangelstatusManagerImpl

```
/*
 * ZWECK: Mangelstatusmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.mangelstatus;
```

```java
import java.util.List;

import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.persister.dao.mangelstatus.MangelstatusDAO;
import
ch.hsluw.mangelmanager.persister.dao.mangelstatus.MangelstatusDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MangelstatusManager zur Verfügung.
 *
 * @version 1.0
 * @author mmont
 *
 */
public class MangelstatusManagerImpl implements MangelstatusManager {

    private MangelstatusDAO mangelstatusDAO = new MangelstatusDAOImpl();

    @Override
    public Mangelstatus add(Mangelstatus entity) throws Exception {

        if (entity.getId() == null) {
            mangelstatusDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Mangelstatus update(Mangelstatus entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return mangelstatusDAO.update(entity);
        }
    }

    @Override
    public void delete(Mangelstatus entity) throws Exception {
        mangelstatusDAO.delete(entity);
    }

    @Override
    public List<Mangelstatus> getAllMangelstatus() throws Exception {
        // TODO Auto-generated method stub
        return mangelstatusDAO.findAllMangelstatus();
    }
}
```

MeldungManager

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */
```

```java
package ch.hsluw.mangelmanager.business.meldung;

import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;



/**
 * Interface fuer Meldung Entity
 *
 * @version 1.0
 * @author cdemir
 *
 */
public interface MeldungManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Meldung add(Meldung entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Meldung update(Meldung entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Meldung entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Meldung findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
```

```
     * @return
     */
    List<Meldung> findAll();

    List<Meldung> findAllMeldungByMangel(Mangel mangel);

}
```

MeldungManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.meldung;

import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.persister.dao.meldung.MeldungDAO;
import ch.hsluw.mangelmanager.persister.dao.meldung.MeldungDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MeldungManager zur Verfügung.
 *
 * @version 1.0
 * @author cdemir
 *
 */
public class MeldungManagerImpl implements MeldungManager {

    private MeldungDAO meldungDAO = new MeldungDAOImpl();

    @Override
    public Meldung add(Meldung entity) throws Exception {

        if (entity.getId() == null) {
            meldungDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Meldung update(Meldung entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return meldungDAO.update(entity);
        }
    }

    @Override
    public void delete(Meldung entity) throws Exception {
```

```
        meldungDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub

    }

    @Override
    public Meldung findById(Integer id) {
        // TODO Auto-generated method stub
        return meldungDAO.findMeldungById(id);
    }

    @Override
    public List<Meldung> findAll() {
        // TODO Auto-generated method stub
        return meldungDAO.findAllMeldung();
    }

    @Override
    public List<Meldung> findAllMeldungByMangel(Mangel mangel) {
        // TODO Auto-generated method stub
        return meldungDAO.findAllMeldungByMangel(mangel);
    }
}
```

MeldungstypManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.meldungstyp;

import java.util.List;

import ch.hsluw.mangelmanager.model.Meldungstyp;



/**
 * Interface fuer Meldungstyp Entity
 *
 * @version 1.0
 * @author cdemir
 *
 */
public interface MeldungstypManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Meldungstyp add(Meldungstyp entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
```

```
     * @param entity
     * @return
     * @throws Exception
     */
    Meldungstyp update(Meldungstyp entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Meldungstyp entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Meldungstyp findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Meldungstyp> findAll();
}
```

MeldungstypManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.meldungstyp;

import java.util.List;

import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.persister.dao.meldungstyp.MeldungstypDAO;
import ch.hsluw.mangelmanager.persister.dao.meldungstyp.MeldungstypDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MeldungstypManager zur Verfügung.
 *
 * @version 1.0
 * @author cdemir
 *
 */
```

```java
public class MeldungstypManagerImpl implements MeldungstypManager {

    private MeldungstypDAO meldungstypDAO = new MeldungstypDAOImpl();

    @Override
    public Meldungstyp add(Meldungstyp entity) throws Exception {

        if (entity.getId() == null) {
            meldungstypDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Meldungstyp update(Meldungstyp entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return meldungstypDAO.update(entity);
        }
    }

    @Override
    public void delete(Meldungstyp entity) throws Exception {
        meldungstypDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub

    }

    @Override
    public Meldungstyp findById(Integer id) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public List<Meldungstyp> findAll() {
        return meldungstypDAO.findAllMeldungstyp();
    }
}
```

ObjekttypManager

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.objekttyp;


import java.util.List;

import ch.hsluw.mangelmanager.model.Objekttyp;
```

```java
/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ObjekttypManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Objekttyp add(Objekttyp entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Objekttyp update(Objekttyp entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Objekttyp entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Objekttyp findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Objekttyp> findAll();

    /**
     * Liefert die Liste mit Objekttypen für die übergebene Bezeichnung
zurück, falls
     * welche gefunden, sonst eine leere Liste.
```

```
     *
     * @param bezeichnung
     * @return
     */
    List<Objekttyp> findByBezeichnung(String bezeichnung);

}
```

ObjekttypManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.objekttyp;
import java.util.List;

import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.persister.dao.objekttyp.ObjekttypDAO;
import ch.hsluw.mangelmanager.persister.dao.objekttyp.ObjekttypDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ObjekttypManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ObjekttypManagerImpl implements ObjekttypManager {

    private ObjekttypDAO objekttypDAO = new ObjekttypDAOImpl();

    @Override
    public Objekttyp add(Objekttyp entity) throws Exception {

        if (entity.getId() == null) {
            objekttypDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Objekttyp update(Objekttyp entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return objekttypDAO.update(entity);
        }
    }

    @Override
```

```
    public void delete(Objekttyp entity) throws Exception {
        objekttypDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub
        objekttypDAO.deleteObjekttypById(id);
    }

    @Override
    public Objekttyp findById(Integer id) {
        return objekttypDAO.findObjekttypById(id);
    }

    @Override
    public List<Objekttyp> findAll() {
        return objekttypDAO.findAllObjekttyp();
    }

    @Override
    public List<Objekttyp> findByBezeichnung(String bezeichnung) {
        return objekttypDAO.findObjekttypByBezeichnung(bezeichnung);
    }


}
```

PersonManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.person;

import java.util.List;

import ch.hsluw.mangelmanager.model.Person;

/**
 * Interface fuer Person Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface PersonManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Person add(Person entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
```

```
     * @return
     * @throws Exception
     */
    Person update(Person entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Person entity) throws Exception;

    /**
     * Löscht die Entity mit übergebener Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Person findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Person> findAll();
}
```

PersonManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.person;
import java.util.List;

import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.persister.dao.person.PersonDAO;
import ch.hsluw.mangelmanager.persister.dao.person.PersonDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * PersonManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
```

```java
public class PersonManagerImpl implements PersonManager {

    private PersonDAO personDAO = new PersonDAOImpl();

    @Override
    public Person add(Person entity) throws Exception {

        if (entity.getId() == null) {
            personDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Person update(Person entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return personDAO.update(entity);
        }
    }

    @Override
    public void delete(Person entity) throws Exception {
        personDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        personDAO.deletePersonById(id);

    }

    @Override
    public Person findById(Integer id) {
        return personDAO.findPersonById(id);
    }

    @Override
    public List<Person> findAll() {
        return personDAO.findAllPerson();
    }
}
```

PlzManager

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.plz;

import java.util.List;

import ch.hsluw.mangelmanager.model.Plz;
```

```java
/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface PlzManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Plz add(Plz entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Plz update(Plz entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Plz entity) throws Exception;

    /**
     * Löscht die Entity mit der übegebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Plz findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Plz> findAll();
}
```

PlzManagerImpl

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.plz;
import java.util.List;

import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.persister.dao.plz.PlzDAO;
import ch.hsluw.mangelmanager.persister.dao.plz.PlzDAOImpl;




/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * PlzManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class PlzManagerImpl implements PlzManager {

    private PlzDAO plzDAO = new PlzDAOImpl();

    @Override
    public Plz add(Plz entity) throws Exception {

        if (entity.getPlz() == null) {
            plzDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getPlz() + ")");
        }
    }

    @Override
    public Plz update(Plz entity) throws Exception {

        if (entity.getPlz() == null) {
            return add(entity);
        } else {
            return plzDAO.update(entity);
        }
    }

    @Override
    public void delete(Plz entity) throws Exception {
        plzDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub
        plzDAO.deletePlzById(id);
    }

    @Override
```

```
    public Plz findById(Integer id) {
        return plzDAO.findPlzById(id);
    }

    @Override
    public List<Plz> findAll() {
        return plzDAO.findAllPlz();
    }



}
```

ProjektManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projekt;

import java.util.List;

import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Projekt;

/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ProjektManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Projekt add(Projekt entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Projekt update(Projekt entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Projekt entity) throws Exception;

    /**
```

```
     * Löscht die Entity mit übergebener Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Projekt findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Projekt> findAll();

    /**
     * Liefert die Liste mit Projekten für die übergebene Bezeichnung
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    List<Projekt> findByBezeichnung(String bezeichnung);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Projektstatus
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param projektstatus
     * @return
     */
    List<Projekt> findByProjektstatus(String projektstatus);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Ort zurück,
falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param ort
     * @return
     */
    List<Projekt> findByOrt(String ort);

    /**
     * Liefert die Liste mit Projekten für die übergebene Plz zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param plz
     * @return
     */
    List<Projekt> findByPlz(String plz);

    /**
```

```
     * Liefert die Liste mit Projekten für den übergebenen Bauherren
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param bauherr
     * @return
     */
    List<Projekt> findByBauherr(String bauherr);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Objekttyp
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param objekttyp
     * @return
     */
    List<Projekt> findByObjekttyp(String objekttyp);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Arbeitstyp
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param arbeitstyp
     * @return
     */
    List<Projekt> findByArbeitstyp(String arbeitstyp);


    List<Projekt> findAllSubunternehmenProjekt(Integer subunternehmen2);

    List<Projekt> findProjektbyPerson(Person person);



}
```

ProjektManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projekt;
import java.util.List;

import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.persister.dao.projekt.ProjektDAO;
import ch.hsluw.mangelmanager.persister.dao.projekt.ProjektDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
```

```
 *
 */
public class ProjektManagerImpl implements ProjektManager {

    private ProjektDAO projektDAO = new ProjektDAOImpl();

    @Override
    public Projekt add(Projekt entity) throws Exception {

        if (entity.getId() == null) {
            projektDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Projekt update(Projekt entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return projektDAO.update(entity);
        }
    }

    @Override
    public void delete(Projekt entity) throws Exception {
        projektDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        projektDAO.deleteProjektById(id);

    }

    @Override
    public Projekt findById(Integer id) {
        return projektDAO.findProjektById(id);
    }

    @Override
    public List<Projekt> findAll() {
        return projektDAO.findAllProjekt();
    }

    @Override
    public List<Projekt> findByBezeichnung(String bezeichnung) {
        return projektDAO.findProjektByBezeichnung(bezeichnung);
    }

    @Override
    public List<Projekt> findByProjektstatus(String projektstatus) {
        return projektDAO.findProjektByProjektstatus(projektstatus);
    }

    @Override
    public List<Projekt> findByOrt(String ort) {
        return projektDAO.findProjektByOrt(ort);
```

```
    }

    @Override
    public List<Projekt> findByPlz(String plz) {
        return projektDAO.findProjektByPlz(plz);
    }

    @Override
    public List<Projekt> findByBauherr(String bauherr) {
        return projektDAO.findProjektByBauherr(bauherr);
    }

    @Override
    public List<Projekt> findByObjekttyp(String objekttyp) {
        return projektDAO.findProjektByObjekttyp(objekttyp);
    }

    @Override
    public List<Projekt> findByArbeitstyp(String arbeitstyp) {
        return projektDAO.findProjektByObjekttyp(arbeitstyp);
    }


    @Override
    public List<Projekt> findAllSubunternehmenProjekt(Integer
subunternehmen) {
        // TODO Auto-generated method stub
        return projektDAO.findAllSubunternehmenProjekt(subunternehmen);
    }

    @Override
    public List<Projekt> findProjektbyPerson(Person person) {
        // TODO Auto-generated method stub
        return projektDAO.findProjektByPerson(person);
    }



}
```

ProjektGuMitarbeiterManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projektgumitarbeiter;

import java.util.List;

import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;


/**
 * Interface fuer ProjektGuMitarbeiterGuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
```

```
public interface ProjektGuMitarbeiterManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    ProjektGuMitarbeiter add(ProjektGuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    ProjektGuMitarbeiter update(ProjektGuMitarbeiter entity) throws
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(ProjektGuMitarbeiter entity) throws Exception;

    void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception;

    /**
     * Liefert die Entity für die übergebenen Werte zurück.
     *
     * @param id
     * @return
     */
    ProjektGuMitarbeiter findById(Integer idProjekt, Integer idMitarbeiter)
throws Exception;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<ProjektGuMitarbeiter> findAll();

    List<ProjektGuMitarbeiter> findAllBauleiterByProjekt(Projekt projekt2)
throws Exception;
}
```

ProjektGuMitarbeiterManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projektgumitarbeiter;
import java.util.List;

import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
```

```java
import
ch.hsluw.mangelmanager.persister.dao.projektgumitarbeiter.ProjektGuMitarbei
terDAO;
import
ch.hsluw.mangelmanager.persister.dao.projektgumitarbeiter.ProjektGuMitarbei
terDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektGuMitarbeiterManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ProjektGuMitarbeiterManagerImpl implements
ProjektGuMitarbeiterManager {

    private ProjektGuMitarbeiterDAO projektGuMitarbeiterDAO = new
ProjektGuMitarbeiterDAOImpl();

    @Override
    public ProjektGuMitarbeiter add(ProjektGuMitarbeiter entity) throws
Exception {

            projektGuMitarbeiterDAO.save(entity);
            return entity;

    }

    @Override
    public ProjektGuMitarbeiter update(ProjektGuMitarbeiter entity) throws
Exception {

        if (entity.getFkMitarbeiter() == null && entity.getFkProjekt() ==
null) {
            return add(entity);
        } else {
            return projektGuMitarbeiterDAO.update(entity);
        }
    }

    @Override
    public void delete(ProjektGuMitarbeiter entity) throws Exception {
        projektGuMitarbeiterDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception {
        projektGuMitarbeiterDAO.deleteProjektGuMitarbeiterById(idProjekt,
idMitarbeiter);

    }

    @Override
    public ProjektGuMitarbeiter findById(Integer idProjekt, Integer
idMitarbeiter) {
        return
projektGuMitarbeiterDAO.findProjektGuMitarbeiterById(idProjekt,
idMitarbeiter);
```

```
    }

    @Override
    public List<ProjektGuMitarbeiter> findAll() {
        return projektGuMitarbeiterDAO.findAllProjektGuMitarbeiter();
    }

    @Override
    public List<ProjektGuMitarbeiter> findAllBauleiterByProjekt(Projekt
projekt2)
            throws Exception {
        // TODO Auto-generated method stub
        return projektGuMitarbeiterDAO.findAllBauleiterByProjekt(projekt2);
    }
}
```

ProjektstatusManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projektstatus;

import java.util.List;

import ch.hsluw.mangelmanager.model.Projektstatus;


/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ProjektstatusManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Projektstatus add(Projektstatus entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Projektstatus update(Projektstatus entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
```

```
    void delete(Projektstatus entity) throws Exception;

    /**
     * Löscht die Entity mit der übegebenen Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Projektstatus findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Projektstatus> findAll();
}
```

ProjektstatusManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projektstatus;
import java.util.List;

import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.persister.dao.projektstatus.ProjektstatusDAO;
import
ch.hsluw.mangelmanager.persister.dao.projektstatus.ProjektstatusDAOImpl;




/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektstatusManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ProjektstatusManagerImpl implements ProjektstatusManager {

    private ProjektstatusDAO projektstatusDAO = new ProjektstatusDAOImpl();

    @Override
    public Projektstatus add(Projektstatus entity) throws Exception {

        if (entity.getId() == null) {
```

```
            projektstatusDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Projektstatus update(Projektstatus entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return projektstatusDAO.update(entity);
        }
    }

    @Override
    public void delete(Projektstatus entity) throws Exception {
        projektstatusDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub
        projektstatusDAO.deleteProjektstatusById(id);
    }

    @Override
    public Projektstatus findById(Integer id) {
        return projektstatusDAO.findProjektstatusById(id);
    }

    @Override
    public List<Projektstatus> findAll() {
        return projektstatusDAO.findAllProjektstatus();
    }



}
```

ProjektSuMitarbeiterManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projektsumitarbeiter;

import java.util.List;

import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;


/**
 * Interface fuer ProjektSuMitarbeiterSuMitarbeiter Entity
 *
 * @version 1.0
```

```
 * @author lkuendig
 *
 */
public interface ProjektSuMitarbeiterManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    ProjektSuMitarbeiter add(ProjektSuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    ProjektSuMitarbeiter update(ProjektSuMitarbeiter entity) throws
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(ProjektSuMitarbeiter entity) throws Exception;

    void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception;

    /**
     * Liefert die Entity für die übergebenen Werte zurück.
     *
     * @param id
     * @return
     */
    ProjektSuMitarbeiter findById(Integer idProjekt, Integer
idMitarbeiter);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<ProjektSuMitarbeiter> findAll();
}
```

ProjektSuMitarbeiterManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.projektsumitarbeiter;
import java.util.List;

import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
```

```java
import
ch.hsluw.mangelmanager.persister.dao.projektsumitarbeiter.ProjektSuMitarbei
terDAO;
import
ch.hsluw.mangelmanager.persister.dao.projektsumitarbeiter.ProjektSuMitarbei
terDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektSuMitarbeiterManager zur Verfüsung.
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class ProjektSuMitarbeiterManagerImpl implements
ProjektSuMitarbeiterManager {

    private ProjektSuMitarbeiterDAO projektSuMitarbeiterDAO = new
ProjektSuMitarbeiterDAOImpl();

    @Override
    public ProjektSuMitarbeiter add(ProjektSuMitarbeiter entity) throws
Exception {
            projektSuMitarbeiterDAO.save(entity);
            return entity;

    }

    @Override
    public ProjektSuMitarbeiter update(ProjektSuMitarbeiter entity) throws
Exception {

        if (entity.getFkMitarbeiter() == null && entity.getFkProjekt() ==
null) {
            return add(entity);
        } else {
            return projektSuMitarbeiterDAO.update(entity);
        }
    }

    @Override
    public void delete(ProjektSuMitarbeiter entity) throws Exception {
        projektSuMitarbeiterDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception {
        projektSuMitarbeiterDAO.deleteProjektSuMitarbeiterById(idProjekt,
idMitarbeiter);

    }

    @Override
    public ProjektSuMitarbeiter findById(Integer idProjekt, Integer
idMitarbeiter) {
        return
projektSuMitarbeiterDAO.findProjektSuMitarbeiterById(idProjekt,
idMitarbeiter);
    }
```

```
    @Override
    public List<ProjektSuMitarbeiter> findAll() {
        return projektSuMitarbeiterDAO.findAllProjektSuMitarbeiter();
    }
}
```

RolleManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.rolle;

import java.util.List;

import ch.hsluw.mangelmanager.model.Rolle;

/**
 * Interface fuer Rolle Entity
 *
 * @version 1.0
 * @author miten
 *
 */
public interface RolleManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Rolle add(Rolle entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Rolle update(Rolle entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Rolle entity) throws Exception;

    /**
     * Löscht die Entity mit übergebener Id.
     *
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;
```

```
    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Rolle findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Rolle> findAll();
}
```

RolleManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.rolle;
import java.util.List;

import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.persister.dao.rolle.RolleDAO;
import ch.hsluw.mangelmanager.persister.dao.rolle.RolleDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * RolleManager zur Verfügung.
 *
 * @version 1.0
 * @author miten
 *
 */
public class RolleManagerImpl implements RolleManager {

    private RolleDAO rolleDAO = new RolleDAOImpl();

    @Override
    public Rolle add(Rolle entity) throws Exception {

        if (entity.getId() == null) {
            rolleDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public Rolle update(Rolle entity) throws Exception {

        if (entity.getId() == null) {
```

```
            return add(entity);
        } else {
            return rolleDAO.update(entity);
        }
    }

    @Override
    public void delete(Rolle entity) throws Exception {
        rolleDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        rolleDAO.deleteRolleById(id);
    }

    @Override
    public Rolle findById(Integer id) {
        return rolleDAO.findRolleById(id);
    }

    @Override
    public List<Rolle> findAll() {
        return rolleDAO.findAllRolle();
    }
}
```

SubunternehmenManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.subunternehmen;

import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * Interface fuer Subunternehmen Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface SubunternehmenManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Subunternehmen add(Subunternehmen entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
```

```
     * @return
     * @throws Exception
     */
    void update(Subunternehmen entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Subunternehmen entity) throws Exception;

    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Subunternehmen findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<Subunternehmen> findAll();


    String findAllProjekte(int subunternehmen);

    void save(Subunternehmen subunternehmen);

    List<SuMitarbeiter> findAllSubunternehmenMitarbeiter(Subunternehmen
subunternehmen);

    List<Subunternehmen> findAllSubunternehmenByProjekt(Integer projekt2);
}


SubunternehmenManagerImpl
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.subunternehmen;
import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
import
ch.hsluw.mangelmanager.persister.dao.subunternehmen.SubunternehmenDAO;
import
ch.hsluw.mangelmanager.persister.dao.subunternehmen.SubunternehmenDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * SubunternehmenManager zur Verfügung.
```

```
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class SubunternehmenManagerImpl implements SubunternehmenManager {

    private SubunternehmenDAO subunternehmenDAO = new
SubunternehmenDAOImpl();

    @Override
    public Subunternehmen add(Subunternehmen entity) throws Exception {

        if (entity.getId() == null) {
            subunternehmenDAO.save(entity);
            return entity;
        } else {
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public void update(Subunternehmen entity) throws Exception {

        if (entity.getId() == null) {
            add(entity);
        } else {
            subunternehmenDAO.update(entity);
        }
    }

    @Override
    public void delete(Subunternehmen entity) throws Exception {
        subunternehmenDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        subunternehmenDAO.deleteSubunternehmenById(id);

    }

    @Override
    public Subunternehmen findById(Integer id) {
        return subunternehmenDAO.findSubunternehmenById(id);
    }

    @Override
    public List<Subunternehmen> findAll() {
        return subunternehmenDAO.findAllSubunternehmen();
    }

    @Override
    public String findAllProjekte(int subunternehmen) {
        return subunternehmenDAO.findAllProjekte(subunternehmen);
    }

    @Override
    public void save(Subunternehmen subunternehmen) {
        try {
            subunternehmenDAO.save(subunternehmen);
```

```
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    @Override
    public List<SuMitarbeiter>
findAllSubunternehmenMitarbeiter(Subunternehmen subunternehmen) {
        // TODO Auto-generated method stub
        return
subunternehmenDAO.findAllSubunternehmenMitarbeiter(subunternehmen);
    }

    @Override
    public List<Subunternehmen> findAllSubunternehmenByProjekt(Integer
projekt2) {
        // TODO Auto-generated method stub
        return subunternehmenDAO.findAllSubunternehmenByProjekt(projekt2);
    }
}
```

SuMitarbeiterManager

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.sumitarbeiter;

import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;

/**
 * Interface fuer SuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface SuMitarbeiterManager {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    SuMitarbeiter add(SuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    SuMitarbeiter update(SuMitarbeiter entity) throws Exception;
```

```
    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(SuMitarbeiter entity) throws Exception;

    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    SuMitarbeiter findById(Integer id);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<SuMitarbeiter> findAll();

}
```

SuMitarbeiterManagerImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.business.sumitarbeiter;
import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.persister.dao.sumitarbeiter.SuMitarbeiterDAO;
import
ch.hsluw.mangelmanager.persister.dao.sumitarbeiter.SuMitarbeiterDAOImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * SuMitarbeiterManager zur Verfügung.
 *
 * @version 1.0
 * @author sritz
 *
 */
public class SuMitarbeiterManagerImpl implements SuMitarbeiterManager {

    private SuMitarbeiterDAO suMitarbeiterDAO = new SuMitarbeiterDAOImpl();

    @Override
    public SuMitarbeiter add(SuMitarbeiter entity) throws Exception {

        if (entity.getId() == null) {
            suMitarbeiterDAO.save(entity);
            return entity;
        } else {
```

```
            throw new Exception(
                    "Entity im Datenbestand bereits vorhanden (Id = "
                            + entity.getId() + ")");
        }
    }

    @Override
    public SuMitarbeiter update(SuMitarbeiter entity) throws Exception {

        if (entity.getId() == null) {
            return add(entity);
        } else {
            return suMitarbeiterDAO.update(entity);
        }
    }

    @Override
    public void delete(SuMitarbeiter entity) throws Exception {
        suMitarbeiterDAO.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        suMitarbeiterDAO.deleteSuMitarbeiterById(id);

    }

    @Override
    public SuMitarbeiter findById(Integer id) {
        return suMitarbeiterDAO.findSuMitarbeiterById(id);
    }

    @Override
    public List<SuMitarbeiter> findAll() {
        return suMitarbeiterDAO.findAllSuMitarbeiter();
    }
}
```

ProjektTest

```
package ch.hsluw.mangelmanager.business;


import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;

import org.junit.Test;

import ch.hsluw.mangelmanager.business.projekt.ProjektManager;
import ch.hsluw.mangelmanager.business.projekt.ProjektManagerImpl;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.Projektstatus;


/**
 * Interface fuer Projekt Entity
```

```
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ProjektTest {

    @Test
    public void test() {
        Adresse a = new Adresse("Haus 12", new Plz(3984, "Fieschertal"));
        Bauherr b = new Bauherr("Schmid", "Hans", "027 971 40 24", a);
        Projektstatus ps = new Projektstatus("offen");
        Arbeitstyp at = new Arbeitstyp("Neubau");
        Objekttyp ot = new Objekttyp("Block");

        List<Bauherr> bauherren = new ArrayList<Bauherr>();

        bauherren.add(b);

        Projekt projekt2 = new Projekt(a, "Neubau Haus Romantica",
bauherren, new GregorianCalendar(2015, 4, 16),
                new GregorianCalendar(2015, 4, 20),ot, at,  new
GregorianCalendar(2015, 4, 20), ps);

        ProjektManager pm = new ProjektManagerImpl();

        try {
            pm.add(projekt2);
            projekt2.setBezeichnung("Block Romantica");
            pm.update(projekt2);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }



        List<Projekt> projekte = pm.findAll();
        System.out.println(projekte.get(0).getEndDatum().getTime());
    }

}
```

ClientWS

```
package ch.hsluw.mangelmanager.client.extern;

import java.net.URL;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import ch.hsluw.mangelmanager.webservice.MangelManagerService;

/**
 * Diese Klasse stellt das Userinterface fuer den MangelMangager via
SOAP/WS
 * zur Verfuegung
 *
 * @version 1.0
 * @author lkuendig
```

```
 *
 */
public class ClientWS {
    public Service service;
    public MangelManagerService proxy;
    private static ClientWS instance;

    public static ClientWS getInstance () {
        if (ClientWS.instance == null) {

                try {
                    ClientWS.instance = new ClientWS();
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            return ClientWS.instance;
      }
    /**
     * Instantiates a new mangelmanager client rmi.
     *
     * @throws Exception
     */
    public ClientWS() throws Exception {

        ///1st argument service URI, refer to wsdl document above
        URL url = new
URL("http://localhost:8080/MangelManager/mangelmanager?wsdl");
        //2nd argument is service name, refer to wsdl document above
        QName qname = new
QName("http://webservice.mangelmanager.hsluw.ch/",
"MangelManagerServiceImplService");

       service = Service.create(url, qname);
        proxy = service.getPort(MangelManagerService.class);
    }
}
```

Main

```
package ch.hsluw.mangelmanager.client.extern;


import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import ch.hsluw.mangelmanager.model.Login;

public class Main extends Application {

    // RMI Client to interact
        ClientWS client = null;
```

```
private static Stage loginStage;
private static AnchorPane loginLayout;
public static Integer loginId;

@FXML
private Label lblLoginError;
@FXML
private TextField txtBenutzer;
@FXML
private PasswordField pwPasswort;
@FXML
private TextField txtIp;
@FXML
private TextField txtPort;

@Override
public void start(Stage primaryStage) {
    this.loginStage = primaryStage;
    this.loginStage.setTitle("Mängelmanager");
    initRootLayout();

}

/**
 * Initializes the demo  layout.
 */
public static void initRootLayout() {
    try {
        // Load root layout from fxml file.
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(Main.class
                .getResource("view/login/Login.fxml"));
        loginLayout = (AnchorPane) loader.load();

        // Show the scene containing the root layout.
        Scene scene = new Scene(loginLayout);
        loginStage.setScene(scene);
        loginStage.show();
        loginStage.setResizable(false);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@FXML
public void login() throws IOException {

    client = ClientWS.getInstance();
    Login data = client.proxy.getLoginByName(txtBenutzer.getText());


    if (data != null){
        if ((txtBenutzer.getText().equals(data.getBenutzername()))
                && (pwPasswort.getText().equals(data.getPasswort()))) {
            lblLoginError.setText("Login erfolgreich!");
            loginId = data.getId();
            Stage stage = new Stage();
            Parent root = FXMLLoader.load(getClass().getResource(
                    "view/root/Root.fxml"));

            Scene scene = new Scene(root);
            stage.setScene(scene);
```

```
                stage.setTitle("Mängelmanager");
                stage.setMaximized(true);
                stage.show();

                Stage stageToClose = (Stage)
txtBenutzer.getScene().getWindow();
                stageToClose.close();
            } else {
                lblLoginError.setText("Fehler beim Einloggen!");
            }
        } else {
            lblLoginError.setText("Fehler beim Einloggen!");
        }

    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

AddMangelController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.DatePicker;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Projekt;

public class AddMangelController implements Initializable {
        //WS Client to interact
        ClientWS client = null;
        RootController rootController = null;
        Projekt projekt = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        Meldung meldung = null;
        Mangel mangel = null;
        Mangelstatus mangelstatus = null;
        Login login = null;
```

```
        List<Mangelstatus> mangelstatusl = null;
        List<Meldungstyp> meldungstypl = null;
        Meldungstyp meldungstyp = null;


        @FXML
        public TextField txtMangelBezeichung;
        @FXML
        public TextArea txtMangelBeschreibung;
        @FXML
        public DatePicker dateMangelFaellig;


        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            // TODO Auto-generated method stub

        }

        public void init(Projekt projekt) {
            try {
                client = new ClientWS();

                this.projekt = projekt;
                mangelstatusl = client.proxy.getAllMangelStatus();
                for (Mangelstatus mangelstatus : mangelstatusl) {
                    if(mangelstatus.getBezeichnung().equals("Offen")){
                        this.mangelstatus = mangelstatus;
                    }
                }
                meldungstypl = client.proxy.getAllMeldungstyp();
                for (Meldungstyp meldungstyp : meldungstypl) {
                    if(meldungstyp.getBezeichnung().equals("Reklamation")){
                        this.meldungstyp = meldungstyp;
                    }
                }
                login = client.proxy.getLoginById(Main.loginId);
                System.out.println(login.getBenutzername()+ " " +
login.getEmail());
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        @FXML
        private void mangelSave() {
            Calendar cl = Calendar.getInstance();

            mangel = new Mangel(projekt,
txtMangelBezeichung.getText(),(GregorianCalendar) cl, new
GregorianCalendar(dateMangelFaellig.getValue().getYear(),
dateMangelFaellig.getValue().getMonthValue()-1,
dateMangelFaellig.getValue().getDayOfMonth()),mangelstatus ,login ,
txtMangelBeschreibung.getText());

            client.proxy.addMangel(mangel);

            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
```

```
                        .getResource("view/mangel/AussererMangel.fxml"));
                AnchorPane mangel = (AnchorPane) loader.load();

                MangelController mangelController =
loader.<MangelController>getController();
                mangelController.setRootController(rootController);

                rootController.rootLayout.setCenter(mangel);

            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        @FXML
        public void addMangelCancel(){
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/mangel/AussererMangel.fxml"));
                AnchorPane mangel = (AnchorPane) loader.load();

                MangelController mangelController =
loader.<MangelController>getController();
                mangelController.setRootController(rootController);

                rootController.rootLayout.setCenter(mangel);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

AddMeldungController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;

public class AddMeldungController implements Initializable {
```

```
        //WS Client to interact
        ClientWS client = null;
        RootController rootController = null;
        Mangel mangel = null;
        GregorianCalendar timestamp = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        Login login = null;


        @FXML
        public Label lblMeldungProjekt;
        @FXML
        private Label lblMeldungMangel;
        @FXML
        private ChoiceBox<Meldungstyp> cbMeldungstyp;
        @FXML
        private TextArea txtMeldungBeschreibung;


        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            // TODO Auto-generated method stub
            try {
                client = new ClientWS();
                for (Meldungstyp meldungstyp :
client.proxy.getAllMeldungstyp()) {
                    cbMeldungstyp.getItems().add(meldungstyp);
                }



            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        public void init(Mangel mangel) {
            try {
                this.mangel = mangel;

lblMeldungProjekt.setText(mangel.getFkProjekt().getBezeichnung());
                lblMeldungMangel.setText(mangel.getBezeichnung());
                login = client.proxy.getLoginById(Main.loginId);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        @FXML
        public void meldungSave() {
            timestamp = (GregorianCalendar) Calendar.getInstance();
            client.proxy.addMeldung(new Meldung(mangel,
cbMeldungstyp.getSelectionModel().getSelectedItem(),
txtMeldungBeschreibung.getText(),timestamp, false, login));
```

```
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        @FXML
        public void addMeldungCancel(){
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

AddPersonController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.TitledPane;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Bauherr;
```

```java
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The AddPersonController handles all interaction with person *
 *
 * @author sritz
 * @version 1.0
 *
 */

public class AddPersonController implements Initializable {
    //WS Client to interact
    ClientWS client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Bauherr bauherr;
    GuMitarbeiter guMitarbeiter;
    SuMitarbeiter suMitarbeiter;
    Login login;
    Adresse adresse;

    //Declare FXML objects
    @FXML
    private TextField txtPersonName;
    @FXML
    private TextField txtPersonVorname;
    @FXML
    private TextField txtPersonStrasse;
    @FXML
    private TextField txtPersonTelefon;
    @FXML
    private Label lblPersonOrt;
    @FXML
    private ComboBox<Plz> cbPersonPlz;
    @FXML
    private ComboBox cbPersonFunktion;
    @FXML
    private ChoiceBox<Subunternehmen> cbPersonUnternehmen;
    @FXML
    private Label lblPersonError;
    @FXML
    private TitledPane tpPersonLogin;
    @FXML
    private TextField txtPersonBenutzername;
    @FXML
    private TextField txtPersonPasswort;
    @FXML
    private TextField txtPersonPasswortWiederholen;
    @FXML
    private TextField txtPersonEmail;
    @FXML
    private ComboBox<Rolle> cbPersonRolle;
```

```
    //Saves a new Person objekt
    @FXML
    public void personSave(){
        // TODO
        if(cbPersonFunktion.getValue() != null){
            if(cbPersonFunktion.getValue().equals("GuMitarbeiter")){

if(txtPersonPasswort.getText().equals(txtPersonPasswortWiederholen.getText(
))){
                    login = new Login(txtPersonBenutzername.getText(),
txtPersonPasswort.getText(), txtPersonEmail.getText(),
cbPersonRolle.getSelectionModel().getSelectedItem());
                    guMitarbeiter = new
GuMitarbeiter(txtPersonName.getText(), txtPersonVorname.getText(),
txtPersonTelefon.getText(), login);
                    client.proxy.addGuMitarbeiter(guMitarbeiter);
                }else{
                    lblPersonError.setText("Passwort wiederholen!");
                    return;
                }
            }else if(cbPersonFunktion.getValue().equals("SuMitarbeiter")){

if(txtPersonPasswort.getText().equals(txtPersonPasswortWiederholen.getText(
))){
                    login = new Login(txtPersonBenutzername.getText(),
txtPersonPasswort.getText(), txtPersonEmail.getText(),
cbPersonRolle.getSelectionModel().getSelectedItem());
                    suMitarbeiter = new
SuMitarbeiter(txtPersonName.getText(), txtPersonVorname.getText(),
txtPersonTelefon.getText(),
cbPersonUnternehmen.getSelectionModel().getSelectedItem(), login);
                    client.proxy.addSuMitarbeiter(suMitarbeiter);
                }else{
                    lblPersonError.setText("Passwort wiederholen!");
                    return;
                }
            }else{
                adresse = new Adresse(txtPersonStrasse.getText(),
cbPersonPlz.getSelectionModel().getSelectedItem());
                bauherr = new Bauherr(txtPersonName.getText(),
txtPersonVorname.getText(), txtPersonTelefon.getText(), adresse);
                client.proxy.addBauherr(bauherr);
            }
        }else{
            lblPersonError.setText("Funktion Auswählen");
            return;
        }
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
            AnchorPane person = (AnchorPane) loader.load();

            PersonController personController =
loader.<PersonController>getController();
            personController.setRootController(rootController);

            rootController.rootLayout.setCenter(person);

        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    }

    @FXML
    public void personCancel(){
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
            AnchorPane person = (AnchorPane) loader.load();

            PersonController personController =
loader.<PersonController>getController();
            personController.setRootController(rootController);

            rootController.rootLayout.setCenter(person);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //enabled or disabled fields if not needed
    @FXML
    public void enableFields(){
        if(cbPersonFunktion.getValue() != null){
            if(cbPersonFunktion.getValue().equals("GuMitarbeiter")){
                tpPersonLogin.setDisable(false);
                cbPersonUnternehmen.setDisable(true);
                cbPersonPlz.setDisable(true);
                txtPersonStrasse.setDisable(true);

            }
            else if(cbPersonFunktion.getValue().equals("SuMitarbeiter")){
                tpPersonLogin.setDisable(false);
                cbPersonUnternehmen.setDisable(false);
                cbPersonPlz.setDisable(true);
                txtPersonStrasse.setDisable(true);
            }
            else if(cbPersonFunktion.getValue().equals("Bauherr")){
                tpPersonLogin.setDisable(true);
                cbPersonUnternehmen.setDisable(true);
                cbPersonPlz.setDisable(false);
                txtPersonStrasse.setDisable(false);
            }
        }
    }



    @FXML
    private void plzChange(){
        if (cbPersonPlz.getSelectionModel().getSelectedItem() != null){

lblPersonOrt.setText(cbPersonPlz.getSelectionModel().getSelectedItem().getO
rt());
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
try {
        client = ClientWS.getInstance();
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }

        for (Plz plz :
FXCollections.observableArrayList(client.proxy.getAllPlz())) {
            cbPersonPlz.getItems().add(plz);
        }
        for (Subunternehmen subunternehmen :
FXCollections.observableArrayList(client.proxy.getAllSubunternehmen())) {
            cbPersonUnternehmen.getItems().add(subunternehmen);
        }
        for (Rolle rolle :
FXCollections.observableArrayList(client.proxy.getAllRolle())) {
            cbPersonRolle.getItems().add(rolle);
        }

cbPersonFunktion.setItems(FXCollections.observableArrayList("Bauherr","SuMi
tarbeiter","GuMitarbeiter"));
    }


}
```

AddProjektController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.Projektstatus;

public class AddProjektController implements Initializable {
        //WS Client to interact
        ClientWS client = null;
```

```
        RootController rootController = null;
        Projekt projekt = null;
        DateFormat formatDatum = null;
        DateTimeFormatter dateFormatter = null;
        List<Bauherr> b = null;
        Adresse a = null;
        Projektstatus ps = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }
        @FXML
        private TextField txtProjektBezeichnung;
        @FXML
        private ChoiceBox<Bauherr> cbProjektBauherr;
        @FXML
        private TextField txtProjektStrasse;
        @FXML
        private ComboBox<Plz> cbProjektPlz;
        @FXML
        private Label lblProjektOrt;
        @FXML
        private ChoiceBox<Objekttyp> cbProjektObjekttyp;
        @FXML
        private ChoiceBox<Arbeitstyp> cbProjektArbeitstyp;
        @FXML
        private DatePicker dateProjektStartdatum;
        @FXML
        private DatePicker dateProjektFaellig;

        @FXML
        private void addProjekt(){
            b = new ArrayList<Bauherr>();
            b.add(cbProjektBauherr.getSelectionModel().getSelectedItem());

            a = new Adresse(txtProjektStrasse.getText(),
cbProjektPlz.getSelectionModel().getSelectedItem());
            projekt = new Projekt(a,txtProjektBezeichnung.getText(),b, new
GregorianCalendar(dateProjektStartdatum.getValue().getYear(),
dateProjektStartdatum.getValue().getMonthValue() -1,
dateProjektStartdatum.getValue().getDayOfMonth()),null,cbProjektObjekttyp.g
etSelectionModel().getSelectedItem(),cbProjektArbeitstyp.getSelectionModel(
).getSelectedItem(), new
GregorianCalendar(dateProjektFaellig.getValue().getYear(),
dateProjektFaellig.getValue().getMonthValue() -1,
dateProjektFaellig.getValue().getDayOfMonth()),ps);
            //client.addAdresse(a);
            client.proxy.addProjekt(projekt);

            try {
                // Load Projekt overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/AusseresProjekt.fxml"));
                AnchorPane projekte = (AnchorPane) loader.load();

                ProjektController projektController =
loader.<ProjektController>getController();
                projektController.setRootController(rootController);

                rootController.rootLayout.setCenter(projekte);
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        @FXML
        private void cancelProjekt(){
            try {
                // Load Projekt overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/AusseresProjekt.fxml"));
                AnchorPane projekte = (AnchorPane) loader.load();

                ProjektController projektController =
loader.<ProjektController>getController();
                projektController.setRootController(rootController);

                rootController.rootLayout.setCenter(projekte);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @FXML
        private void plzChange(){
            if (cbProjektPlz.getSelectionModel().getSelectedItem() !=
null){

lblProjektOrt.setText(cbProjektPlz.getSelectionModel().getSelectedItem().ge
tOrt());
            }
        }



        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            // TODO Auto-generated method stub
            client = ClientWS.getInstance();
            formatDatum = new SimpleDateFormat("dd.MM.yyyy");
            dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
            for (Bauherr bauherr :
FXCollections.observableArrayList(client.proxy.getAllBauherr())) {
                cbProjektBauherr.getItems().add(bauherr);
            }
            for (Plz plz :
FXCollections.observableArrayList(client.proxy.getAllPlz())) {
                cbProjektPlz.getItems().add(plz);
            }
            for (Objekttyp objekttyp :
FXCollections.observableArrayList(client.proxy.getAllObjekttyp())) {
                cbProjektObjekttyp.getItems().add(objekttyp);
            }
            for (Arbeitstyp arbeitstyp :
FXCollections.observableArrayList(client.proxy.getAllArbeitstyp())) {
                cbProjektArbeitstyp.getItems().add(arbeitstyp);
            }
            for (Projektstatus projektstatus :
FXCollections.observableArrayList(client.proxy.getAllProjektstatus())) {
                if(projektstatus.getBezeichnung().equals("Offen")){
```

```
                    ps = projektstatus;
                }
            }

            // Client interaction
            try {
                client = ClientWS.getInstance();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

}
```

AddUnternehmenController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The AddUnternehmenController handles all interaction with unternehmen *
 *
 * @author lkuendig
 * @version 1.0
 *
 */

public class AddUnternehmenController implements Initializable {
    //WS Client to interact
    ClientWS client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }
    Subunternehmen subunternehmen = null;
    Adresse adresse = null;

    @FXML
    public Label lblUnternehmenId;
    @FXML
    public TextField txtUnternehmenName;
    @FXML
    public TextField txtUnternehmenTelefon;
```

```
    @FXML
    public TextField txtUnternehmenStrasse;
    @FXML
    public ComboBox<Plz> cbUnternehmenPlz;
    @FXML
    public Label lblUnternehmenOrt;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            client = new ClientWS().getInstance();
            for (Plz plz :
FXCollections.observableArrayList(client.proxy.getAllPlz())) {
                cbUnternehmenPlz.getItems().add(plz);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }


    @FXML
    private void plzChange(){
        if (cbUnternehmenPlz.getSelectionModel().getSelectedItem() !=
null){

lblUnternehmenOrt.setText(cbUnternehmenPlz.getSelectionModel().getSelectedI
tem().getOrt());
        }
    }
    @FXML
    private void unternehmenSave() {

        adresse = new
Adresse(txtUnternehmenStrasse.getText(),cbUnternehmenPlz.getSelectionModel(
).getSelectedItem());
        subunternehmen = new Subunternehmen(adresse,
txtUnternehmenName.getText(), txtUnternehmenTelefon.getText());
        client.proxy.addSubunternehmen(subunternehmen);

        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
            subunternehmenController.setRootController(rootController);

            rootController.rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    @FXML
    public void unternehmenCancel(){
        try {
```

```
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
            subunternehmenController.setRootController(rootController);

            rootController.rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }


}
```

MangelController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Mangel;

/**
 * The MangelController handles all interaction with Mangel *
 *
 * @author sritz & mmont
 * @version 1.0
 *
 */
```

```
public class MangelController implements Initializable {
    // WS Client to interact
    ClientWS client = null;

    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    // Define overviewtable with columns
    @FXML
    private TableView<Mangel> tblMangel;
    @FXML
    private TableColumn<Mangel, String> colMangelId;
    @FXML
    private TableColumn<Mangel, String> colMangelBezeichnung;
    @FXML
    private TableColumn<Mangel, String> colMangelProjekt;
    @FXML
    private TableColumn<Mangel, String> colMangelErfassungsdatum;
    @FXML
    private TableColumn<Mangel, String> colMangelFaelligkeitsdatum;
    @FXML
    private TableColumn<Mangel, String> colMangelAbschlusszeit;
    @FXML
    private TableColumn<Mangel, String> colMangelMangelstatus;

    // Datalist for Tableview
    ObservableList<Mangel> data;

    // SetCellValueFactory from overviewtable
    @Override
    public void initialize(URL location, ResourceBundle resources) {

        DateFormat formatDatum = new SimpleDateFormat("dd.MM.yyyy");
        DateFormat formatZeit = new SimpleDateFormat("dd.MM.yyyy hh:mm");

        colMangelId
                .setCellValueFactory(new PropertyValueFactory<Mangel,
String>(
                        "id"));
        colMangelBezeichnung
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getBezeichnung());
                    }
                });
        colMangelProjekt
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkProjekt().getBezeichnung());
                    }
                });

        colMangelErfassungsdatum
```

```
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        if (p.getValue().getErfassungsZeit() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new
SimpleStringProperty(formatZeit.format(p

.getValue().getErfassungsZeit().getTime()));
                        }
                    }
                });

        colMangelFaelligkeitsdatum
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        if (p.getValue().getFaelligkeitsDatum() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new SimpleStringProperty(formatDatum

.format(p.getValue().getFaelligkeitsDatum()
                                                    .getTime()));
                        }
                    }
                });

        colMangelAbschlusszeit
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        if (p.getValue().getAbschlussZeit() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new
SimpleStringProperty(formatZeit.format(p

.getValue().getAbschlussZeit().getTime()));
                        }
                    }
                });

        colMangelMangelstatus
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkMangelstatus().getBezeichnung());
                    }
                });

        // Client interaction
        try {
            client = ClientWS.getInstance();
            data =
FXCollections.observableArrayList(client.proxy.getAllMangel());
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }

        // Set data to tableview
        tblMangel.setItems(data);
    }

    /**
     * prints the TableView into a .csv file
     *
     * @throws IOException
     */
    public void exportTableView() throws IOException {
        DateFormat formatZeit = new SimpleDateFormat("dd.MM.yyyy hh:mm");
        Writer writer = null;
        try {
            client = ClientWS.getInstance();
            data =
FXCollections.observableArrayList(client.proxy.getAllMangel());
            /*
             * Creates a CSV File in which the List will be saved C: is the
             * directory in which the File will be saved
             */
            File file = new File("C:" + "\\" + formatZeit.format(new
Date())
                    + "Mangelliste.csv.");
            writer = new BufferedWriter(new FileWriter(file));

            // The data that has to be put into the .csv File
            for (Mangel mangel : data) {
                String text = mangel.getId()
                        + ";"
                        + mangel.getFkProjekt().getBezeichnung()
                        + ";"
                        + mangel.getBezeichnung()
                        + ";"
                        + formatZeit.format(mangel.getErfassungsZeit()
                                .getTime())
                        + ";"
                        + formatZeit.format(mangel.getFaelligkeitsDatum()
                                .getTime()) + ";"
                        + mangel.getFkMangelstatus().getBezeichnung() + ";"
                        + "\n";

                writer.write(text);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {

            writer.flush();
            writer.close();
        }
    }

    @FXML
    public void showMangelDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {

System.out.println(tblMangel.getSelectionModel().getSelectedItem()
                    .getId());

            try {
```

```
                // Load MangelDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/mangel/InnererMangel.fxml"));
                AnchorPane innererMangel = (AnchorPane) loader.load();

                MangelDetailController detailMangelController = loader
                        .<MangelDetailController> getController();
                detailMangelController.setRootController(rootController);

                detailMangelController.init(tblMangel.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(innererMangel);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

MangelDetailController

```
package ch.hsluw.mangelmanager.client.extern.controller;

    import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;

    public class MangelDetailController implements Initializable {

        //WS Client to interact
            ClientWS client = null;
            RootController rootController = null;
            DateFormat formatDatum = null;
            DateTimeFormatter dateFormatter = null;

            public void setRootController(RootController rootController) {
                // TODO Auto-generated method stub
                this.rootController = rootController;
            }
```

```
            Mangel mangel = null;
            List<Mangelstatus> mangelstatusl = null;
            Mangelstatus mangelstatus = null;

            @FXML
            public Label lblMangelId;
            @FXML
            public TextArea txtMangelBeschreibung;
            @FXML
            public TextField txtMangelDatumanfang;
            @FXML
            public Label lblMangelFaellig;
            @FXML
            public DatePicker dateMangelDatumende;
            @FXML
            public Label lblMangelStatus;
            @FXML
            public Label lblMangelBezeichung;


            public void initialize(URL location, ResourceBundle resources)
{

            }

            public void init(Integer MangelId) {
                    try {
                    client = ClientWS.getInstance();
                    mangel = client.proxy.getMangelById(MangelId);
                    mangelstatusl = client.proxy.getAllMangelStatus();
                    for (Mangelstatus mangelstatus : mangelstatusl) {
                        ;

if(mangelstatus.getBezeichnung().equals("Abgeschlossen")){
                            this.mangelstatus = mangelstatus;
                        }
                    }

                    formatDatum = new SimpleDateFormat("dd.MM.yyyy");
                    dateFormatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy");

                    lblMangelId.setText(mangel.getId().toString());
                    lblMangelBezeichung.setText(mangel.getBezeichnung());

lblMangelFaellig.setText((formatDatum.format(mangel.getFaelligkeitsDatum().
getTime())));

txtMangelBeschreibung.setText(mangel.getBeschreibung());

txtMangelDatumanfang.setText((formatDatum.format(mangel.getErfassungsZeit()
.getTime())));

                    if(mangel.getAbschlussZeit() == null){
                        dateMangelDatumende.setValue(null);
                    }else{

dateMangelDatumende.setValue(LocalDate.parse(formatDatum.format(mangel.getA
bschlussZeit().getTime()), dateFormatter));
                    }

lblMangelStatus.setText(mangel.getFkMangelstatus().getBezeichnung());
```

```
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            @FXML
            private void mangelClose() {
                mangel.setFkMangelstatus(mangelstatus);
                if(dateMangelDatumende.getValue() != null){
                    mangel.setAbschlussZeit(new
GregorianCalendar(dateMangelDatumende.getValue().getYear(),
dateMangelDatumende.getValue().getMonthValue() -1,
dateMangelDatumende.getValue().getDayOfMonth()));
                }
                lblMangelStatus.setText("Abgeschlossen");
                client.proxy.updateMangel(mangel);

            }
            @FXML
            public void mangelCancel(){
                try {
                    // Load Unternehmen overview.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class

.getResource("view/mangel/AussererMangel.fxml"));
                    AnchorPane mangel = (AnchorPane) loader.load();

                    MangelController mangelController =
loader.<MangelController>getController();
                    mangelController.setRootController(rootController);

                    rootController.rootLayout.setCenter(mangel);

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

}
```

MeldungController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
```

```
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Meldung;

/**
 * The MeldungController handles all interaction with meldungen *
 *
 * @author lkuendig
 * @version 1.0
 *
 */

public class MeldungController implements Initializable {

    //WS Client to interact
        ClientWS client = null;

        RootController rootController = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        //Define overviewtable with columns
        @FXML
        private TableView<Meldung> tblMeldung;
        @FXML
        private TableColumn<Meldung, String> colMeldungId;
        @FXML
        private TableColumn<Meldung, String> colMeldungProjekt;
        @FXML
        private TableColumn<Meldung, String> colMeldungMangel;
        @FXML
        private TableColumn<Meldung, String> colMeldungTyp;
        @FXML
        private TableColumn<Meldung, String> colMeldungErfasst;
        @FXML
        private TableColumn<Meldung, String> colMeldungQuittiert;

        //Datalist for Tableview
        ObservableList<Meldung> data;

        //SetCellValueFactory from overviewtable
        @Override
        public void initialize(URL location, ResourceBundle resources) {

            DateFormat formatDatum = new SimpleDateFormat("dd.MM.yyyy");

            colMeldungId.setCellValueFactory(new
PropertyValueFactory<Meldung, String>("id"));
            colMeldungProjekt.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    return new
SimpleStringProperty(p.getValue().getFkMangel().getFkProjekt().getBezeichnu
ng());
                }
```

```
            });
            colMeldungMangel.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    return new
SimpleStringProperty(String.valueOf(p.getValue().getFkMangel().getId()));
                }
            });
            colMeldungTyp.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    return new
SimpleStringProperty(p.getValue().getFkMeldungstyp().getBezeichnung());
                }
            });
            colMeldungErfasst.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    if (p.getValue().getZeitpunkt() == null){
                        return new SimpleStringProperty(" ");
                    }else{
                        return new
SimpleStringProperty(formatDatum.format(p.getValue().getZeitpunkt().getTime
()));
                    }
                }
            });
            colMeldungQuittiert.setCellValueFactory(new
PropertyValueFactory<Meldung, String>("quittiert"));

            //Client interaction
            try {
                client = new ClientWS();
                data =
FXCollections.observableArrayList(client.proxy.getAllMeldung());
            } catch (Exception e) {
                e.printStackTrace();
            }

            //Set data to tableview
            tblMeldung.setItems(data);

        }

        @FXML
        public void showMeldungDetail(MouseEvent t) throws IOException{
            if(t.getClickCount() == 2){

System.out.println(tblMeldung.getSelectionModel().getSelectedItem().getId()
);

                try {
                    // Load MeldungDetail View.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class

.getResource("view/meldung/InnereMeldung.fxml"));
                    AnchorPane inneresMeldung = (AnchorPane) loader.load();
```

```
                    MeldungDetailController detailMeldungController =
loader.<MeldungDetailController>getController();

detailMeldungController.setRootController(rootController);


detailMeldungController.init(tblMeldung.getSelectionModel().getSelectedItem
().getId());
                    rootController.rootLayout.setCenter(inneresMeldung);


                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        }
}
```

MeldungDetailController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Meldung;

public class MeldungDetailController implements Initializable {

    //WS Client to interact
        ClientWS client = null;
        RootController rootController = null;
        DateFormat formatZeit;


        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        Meldung meldung = null;

        @FXML
        public Label lblMeldungId;
        @FXML
        public Label lblMeldungProjekt;
        @FXML
        public Label lblMeldungMangel;
        @FXML
        public Label lblMeldungDatum;
```

```
        @FXML
        public Label lblMeldungArt;
        @FXML
        public TextArea txtMeldungBeschreibung;

        @Override
        public void initialize(URL location, ResourceBundle resources) {
            formatZeit = new SimpleDateFormat("dd.MM.yyyy hh:mm");

        }


        public void init(int meldungId) {
                try {
                client = ClientWS.getInstance();
                meldung = client.proxy.getMeldungById(meldungId);
                lblMeldungId.setText(meldung.getId().toString());

lblMeldungProjekt.setText(meldung.getFkMangel().getFkProjekt().getBezeichnu
ng());

lblMeldungMangel.setText(meldung.getFkMangel().getBezeichnung());

lblMeldungDatum.setText(formatZeit.format(meldung.getZeitpunkt().getTime())
);

lblMeldungArt.setText(meldung.getFkMeldungstyp().getBezeichnung());
                txtMeldungBeschreibung.appendText(meldung.getText());

            } catch (Exception e) {
                e.printStackTrace();
            }

        }

        @FXML
        private void meldungCancel() {
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        @FXML
        public void meldungRead(){
            meldung.setQuittiert(true);
            client.proxy.updateMeldung(meldung);

            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
```

```
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

PersonController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;

/**
 * The PersonController handle all interactions with Persons
 *
 * @author Sandro
 * @version 1.0
 */
public class PersonController implements Initializable {
    RootController rootController = null;
    // WS Client to interact
    ClientWS client = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub

        this.rootController = rootController;
    }
```

```
    // Define overviewtable with columns
    @FXML
    private TableView<Person> tblPerson;
    @FXML
    private TableColumn<Person, String> colPersonId;
    @FXML
    private TableColumn<Person, String> colPersonName;
    @FXML
    private TableColumn<Person, String> colPersonVorname;
    @FXML
    private TableColumn<Person, String> colPersonTyp;
    @FXML
    private TableColumn<Person, String> colPersonUnternehmen;
    @FXML
    private TableColumn<Person, String> colPersonTelefon;
    @FXML
    private TableColumn<Person, String> colPersonBenutzername;
    @FXML
    private TableColumn<Person, String> colPersonEmail;
    @FXML
    private TableColumn<Person, String> colPersonRolle;




    // Datalist for Tableview
    ObservableList<Person> data;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        setCellValueFactoryTblPerson();

        //Client interaction
                try {
                    client = ClientWS.getInstance();
                    data =
FXCollections.observableArrayList(client.proxy.getAllPerson());
                } catch (Exception e) {
                    e.printStackTrace();
                }


                //Set data to tableview
                tblPerson.setItems(data);
    }



    private void setCellValueFactoryTblPerson() {
        colPersonId.setCellValueFactory(new PropertyValueFactory<Person,
String>("id"));
        colPersonName.setCellValueFactory(new PropertyValueFactory<Person,
String>("nachname"));
        colPersonVorname.setCellValueFactory(new
PropertyValueFactory<Person, String>("vorname"));
        colPersonTyp.setCellValueFactory(new
Callback<CellDataFeatures<Person, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(CellDataFeatures<Person,
String> p) {
                if (p.getValue() instanceof Bauherr) {
                     return new SimpleStringProperty("Bauherr");
```

```
                }
                else if (p.getValue() instanceof GuMitarbeiter) {
                     return new SimpleStringProperty("General-
Mitarbeiter");
                }
                else if (p.getValue() instanceof SuMitarbeiter) {
                     return new SimpleStringProperty("Subunternehmen-
Mitarbeiter");
                }
                else{
                    return new SimpleStringProperty("unbekannt");
                }

            }
        });

        colPersonUnternehmen.setCellValueFactory(new
Callback<CellDataFeatures<Person, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(CellDataFeatures<Person,
String> p) {
                if (p.getValue() instanceof Bauherr) {
                     return new SimpleStringProperty("Kein Unternehmen");
                }
                else if (p.getValue() instanceof GuMitarbeiter) {
                     return new SimpleStringProperty("W & W");
                }
                else if (p.getValue() instanceof SuMitarbeiter) {
                     return new SimpleStringProperty(((SuMitarbeiter)
p.getValue()).getFkSubunternehmen().getName());
                }
                else{
                    return new SimpleStringProperty("unbekannt");
                }

            }
        });

        colPersonTelefon.setCellValueFactory(new
PropertyValueFactory<Person, String>("telefon"));
        colPersonBenutzername.setCellValueFactory(new
Callback<CellDataFeatures<Person, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(CellDataFeatures<Person,
String> p) {
                if (p.getValue() instanceof Bauherr) {
                     return new SimpleStringProperty("-");
                }
                else if (p.getValue() instanceof GuMitarbeiter) {
                     return new SimpleStringProperty(((GuMitarbeiter)
p.getValue()).getFkLogin().getBenutzername());
                }
                else if (p.getValue() instanceof SuMitarbeiter) {
                     return new SimpleStringProperty(((SuMitarbeiter)
p.getValue()).getFkLogin().getBenutzername());
                }
                else{
                    return new SimpleStringProperty("unbekannt");
                }
            }
        });
        colPersonEmail.setCellValueFactory(new
Callback<CellDataFeatures<Person, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(CellDataFeatures<Person,
String> p) {
```

```
                if (p.getValue() instanceof Bauherr) {
                     return new SimpleStringProperty("-");
                }
                else if (p.getValue() instanceof GuMitarbeiter) {
                     return new SimpleStringProperty(((GuMitarbeiter)
p.getValue()).getFkLogin().getEmail());
                }
                else if (p.getValue() instanceof SuMitarbeiter) {
                     return new SimpleStringProperty(((SuMitarbeiter)
p.getValue()).getFkLogin().getEmail());
                }
                else{
                    return new SimpleStringProperty("unbekannt");
                }

            }
        });
        colPersonRolle.setCellValueFactory(new
Callback<CellDataFeatures<Person, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(CellDataFeatures<Person,
String> p) {
                if (p.getValue() instanceof Bauherr) {
                     return new SimpleStringProperty("-");
                }
                else if (p.getValue() instanceof GuMitarbeiter) {
                     return new SimpleStringProperty(((GuMitarbeiter)
p.getValue()).getFkLogin().getFkrolle().getName());
                }
                else if (p.getValue() instanceof SuMitarbeiter) {
                     return new SimpleStringProperty(((SuMitarbeiter)
p.getValue()).getFkLogin().getFkrolle().getName());
                }
                else{
                    return new SimpleStringProperty("unbekannt");
                }

            }
        });
    }


    @FXML
    public void showPersonDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {

System.out.println(tblPerson.getSelectionModel().getSelectedItem()
                    .getId());

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/person/InnerePerson.fxml"));
                AnchorPane inneresPerson = (AnchorPane) loader.load();

                PersonDetailController detailPersonController = loader
                        .<PersonDetailController> getController();
                detailPersonController.setRootController(rootController);


detailPersonController.init(tblPerson.getSelectionModel().getSelectedItem()
                        .getId());
                rootController.rootLayout.setCenter(inneresPerson);
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    private void addPerson(){
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AddPerson.fxml"));
            AnchorPane addPerson = (AnchorPane) loader.load();

            AddPersonController addPersonController =
loader.<AddPersonController>getController();
            addPersonController.setRootController(rootController);

            rootController.rootLayout.setCenter(addPerson);


        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @FXML
    private void deletePerson(){

    }
}
```

PersonDetailController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.TitledPane;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
```

```java
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;



/**
 * The ProjektDetailController handles all interaction with person *
 *
 * @author mmont
 * @version 1.0
 *
 */
public class PersonDetailController implements Initializable {
    //WS Client to interact
    ClientWS client = null;
    RootController rootController = null;
    DateFormat formatDatum = null;
    DateTimeFormatter dateFormatter = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Person person = null;

    //Left Panel
    @FXML
    private Label lblPersonId;
    @FXML
    private Label lblPersonName;
    @FXML
    private Label lblPersonVorname;
    @FXML
    private Label lblPersonUnternehmen;
    @FXML
    private TextField txtPersonStrasse;
    @FXML
    private ComboBox<Plz> cbPersonPlz;
    @FXML
    private Label lblPersonOrt;
    @FXML
    private TextField txtPersonTelefon;
    @FXML
    private Label lblPersonUnternehmenanz;

    // Right Panel
    @FXML
    private TitledPane tpPersonLogin;
    @FXML
    private Label lblPersonBenutzername;
    @FXML
    private Label lblPersonLoginRolle;
    @FXML
    private TextField txtPersonEmail;
    @FXML
    private TableView<Projekt> tblPersonProjekt;
```

```
    @FXML
    private TableColumn<Projekt, String> colPersonProjektid;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektbezeichnung;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektbauherr;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektadresse;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektabgeschlossen;

    ObservableList<Projekt> projektData;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            client = ClientWS.getInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }

        formatDatum = new SimpleDateFormat("dd.MM.yyyy");
        dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        for (Plz plz :
FXCollections.observableArrayList(client.proxy.getAllPlz())) {
            cbPersonPlz.getItems().add(plz);
        }

        setCellValueFactoryTblProjekt();
    }

    private void setCellValueFactoryTblProjekt() {
        colPersonProjektid.setCellValueFactory(new
PropertyValueFactory<Projekt, String>("id"));
        colPersonProjektbezeichnung.setCellValueFactory(new
PropertyValueFactory<Projekt, String>("bezeichnung"));
        colPersonProjektbauherr.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(
                    CellDataFeatures<Projekt, String> p) {
                return new SimpleStringProperty(p.getValue()
                        .getFkBauherr().get(0).getNachname()
                        + " "
                        + p.getValue().getFkBauherr().get(0)
                                .getVorname());
            }
        });

        colPersonProjektadresse.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(
                    CellDataFeatures<Projekt, String> p) {
                return new SimpleStringProperty(p.getValue()
                        .getFkAdresse().getStrasse()
                        + " "
                        + p.getValue().getFkAdresse().getPlz().getPlz()
                        + " "
                        + p.getValue().getFkAdresse().getPlz().getOrt());
            }
        });
        colPersonProjektabgeschlossen.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
            public ObservableValue<String> call(
```

```
                    CellDataFeatures<Projekt, String> p) {
                return new SimpleStringProperty(p.getValue()
                        .getFkProjektstatus().getBezeichnung());
            }
        });
    }
    public void init(int personId) {
        try {
        person = client.proxy.getPersonById(personId);
        projektData =
FXCollections.observableArrayList(client.proxy.getProjektbyPerson(person));
        lblPersonId.setText(person.getId().toString());
        lblPersonVorname.setText(person.getVorname());
        lblPersonName.setText(person.getNachname());

        if(person instanceof Bauherr){
            cbPersonPlz.setDisable(false);
            lblPersonOrt.setDisable(false);
            txtPersonStrasse.setDisable(false);

cbPersonPlz.getSelectionModel().select(((Bauherr)person).getFkAdresse().get
Plz().getPlz());

cbPersonPlz.setPromptText(((Bauherr)person).getFkAdresse().getPlz().getPlz(
).toString());
            lblPersonOrt.setText(((Bauherr)
person).getFkAdresse().getPlz().getOrt());

txtPersonStrasse.setText(((Bauherr)person).getFkAdresse().getStrasse());
        }
        txtPersonTelefon.setText(person.getTelefon());
        if(person instanceof GuMitarbeiter || person instanceof
SuMitarbeiter){
            tpPersonLogin.setVisible(true);
            if(person instanceof SuMitarbeiter){
            lblPersonUnternehmenanz.setVisible(true);
            lblPersonUnternehmen.setVisible(true);

lblPersonUnternehmenanz.setText(((SuMitarbeiter)person).getFkSubunternehmen
().getName());

lblPersonBenutzername.setText(((SuMitarbeiter)person).getFkLogin().getBenut
zername());

txtPersonEmail.setText(((SuMitarbeiter)person).getFkLogin().getEmail());

lblPersonLoginRolle.setText(((SuMitarbeiter)person).getFkLogin().getFkrolle
().getName());
            }
            if(person instanceof GuMitarbeiter){

lblPersonBenutzername.setText(((GuMitarbeiter)person).getFkLogin().getBenut
zername());

txtPersonEmail.setText(((GuMitarbeiter)person).getFkLogin().getEmail());

lblPersonLoginRolle.setText(((GuMitarbeiter)person).getFkLogin().getFkrolle
().getName());
                }
        }
        tblPersonProjekt.setItems(projektData);

    } catch (Exception e) {
```

```
            e.printStackTrace();
        }

        }

        @FXML
        public void showProjektDetail(MouseEvent t) throws IOException{
            if (t.getClickCount() == 2) {

System.out.println(tblPersonProjekt.getSelectionModel().getSelectedItem()
                        .getId());

                try {
                    // Load ProjektDetail View.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class
                            .getResource("view/projekt/InneresProjekt.fxml"));
                    AnchorPane inneresProjekt = (AnchorPane) loader.load();

                    ProjektDetailController detailProjektController = loader
                            .<ProjektDetailController> getController();
                    detailProjektController.setRootController(rootController);


detailProjektController.init(tblPersonProjekt.getSelectionModel()
                            .getSelectedItem().getId());
                    rootController.rootLayout.setCenter(inneresProjekt);

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        }

        @FXML
        private void plzChange(){
            if (cbPersonPlz.getSelectionModel().getSelectedItem() != null){

lblPersonOrt.setText(cbPersonPlz.getSelectionModel().getSelectedItem().getO
rt());
            }
        }

        @FXML
        public void personCancel(){
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/person/AusserePerson.fxml"));
                AnchorPane person = (AnchorPane) loader.load();

                PersonController personController =
loader.<PersonController>getController();
                personController.setRootController(rootController);

                rootController.rootLayout.setCenter(person);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

```
    @FXML
    public void personSave(){
        if(person instanceof SuMitarbeiter){
            person.setTelefon(txtPersonTelefon.getText());
            ((SuMitarbeiter)
person).getFkLogin().setEmail(txtPersonEmail.getText());
        }
        else if(person instanceof GuMitarbeiter){
            person.setTelefon(txtPersonTelefon.getText());
            ((GuMitarbeiter)
person).getFkLogin().setEmail(txtPersonEmail.getText());
        }
        else if(person instanceof Bauherr){
            ((Bauherr)
person).getFkAdresse().setStrasse(txtPersonStrasse.getText());
            ((Bauherr)
person).getFkAdresse().setPlz(cbPersonPlz.getSelectionModel().getSelectedIt
em());
            person.setTelefon(txtPersonTelefon.getText());
        }
            client.proxy.updatePerson(person);
    }
}
```

ProjektController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Projekt;

/**
 * The ProjektController handles all interaction with projects *
 *
 * @author sritz
 * @version 1.0
 *
 */
```

```
public class ProjektController implements Initializable {
    // WS Client to interact
    ClientWS client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    // Define overviewtable with columns
    @FXML
    private TableView<Projekt> tblProjekt;
    @FXML
    private TableColumn<Projekt, String> colProjektId;
    @FXML
    private TableColumn<Projekt, String> colProjektBezeichnung;
    @FXML
    private TableColumn<Projekt, String> colProjektBauherr;
    @FXML
    private TableColumn<Projekt, String> colProjektAdresse;
    @FXML
    private TableColumn<Projekt, String> colProjektOffeneMaengel;
    @FXML
    private TableColumn<Projekt, String> colProjektOffeneMeldungen;
    @FXML
    private TableColumn<Projekt, String> colProjektAbgeschlossen;

    // Datalist for Tableview
    ObservableList<Projekt> data;

    // Suche
    @FXML
    private ChoiceBox<String> cbProjektSearch;
    @FXML
    private TextField txtProjektSearch;

    @Override
    public void initialize(URL location, ResourceBundle resources) {

        setCellValueFactoryTblProjekt();

        // Client interaction
        try {
            client = ClientWS.getInstance();
            data =
FXCollections.observableArrayList(client.proxy.getAllProjekt());
        } catch (Exception e) {
            e.printStackTrace();
        }

        cbProjektSearch.getItems().addAll(
                FXCollections.observableArrayList("Bezeichnung", "Bauherr",
                        "Plz", "Ort",
                        "Projektstatus"));
        cbProjektSearch.getSelectionModel().selectFirst();

        //If selected Item is changed clean txtProjektSearch
        cbProjektSearch.valueProperty().addListener(new
ChangeListener<String>() {
            @Override
            public void changed(ObservableValue<? extends String> ov,
String t, String t1) {
```

```
                txtProjektSearch.setText("");
            }
        });
        // Handle TextField text changes.
        txtProjektSearch.textProperty().addListener(new
ChangeListener<String>() {
            @Override
            public void changed(final ObservableValue<? extends String>
observable, final String oldValue, final String newValue) {
                if(newValue.length() <1){

updateTblProjekt(FXCollections.observableArrayList(client.proxy.getAllProje
kt()));
                }
                else{
                switch
(cbProjektSearch.getSelectionModel().getSelectedItem()) {
                    case "Bezeichnung":

updateTblProjekt(FXCollections.observableArrayList(client.proxy.getProjektB
yBezeichnung(txtProjektSearch.getText())));
                        break;
                    case "Bauherr":

updateTblProjekt(FXCollections.observableArrayList(client.proxy.getProjektB
yBauherr(txtProjektSearch.getText())));
                        break;
                    case "Plz":

updateTblProjekt(FXCollections.observableArrayList(client.proxy.getProjektB
yPlz(txtProjektSearch.getText())));
                        break;
                    case "Ort":

updateTblProjekt(FXCollections.observableArrayList(client.proxy.getProjektB
yOrt(txtProjektSearch.getText())));
                        break;
                    case "Projektstatus":

updateTblProjekt(FXCollections.observableArrayList(client.proxy.getProjektB
yProjektstatus(txtProjektSearch.getText())));
                        break;
                    default:
                        updateTblProjekt(data);
                        break;
                    }
                }
            }

        });

        // Set data to tableview
        updateTblProjekt(data);
    }

    private void updateTblProjekt(ObservableList<Projekt> data) {
        // TODO Auto-generated method stub
            tblProjekt.setItems(data);
    }

    private void setCellValueFactoryTblProjekt() {
        // SetCellValueFactory from overviewtable
        colProjektId
```

```
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "id"));
        colProjektBezeichnung
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "bezeichnung"));

        colProjektBauherr
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkBauherr().get(0).getNachname()
                                + " "
                                + p.getValue().getFkBauherr().get(0)
                                        .getVorname());
                    }
                });

        colProjektAdresse
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getStrasse()
                                + " "
                                +
p.getValue().getFkAdresse().getPlz().getPlz()
                                + " "
                                +
p.getValue().getFkAdresse().getPlz().getOrt());
                    }
                });


        colProjektAbgeschlossen
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkProjektstatus().getBezeichnung());
                    }
                });
    }

    @FXML
    public void showProjektDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {

System.out.println(tblProjekt.getSelectionModel().getSelectedItem()
                    .getId());

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/InneresProjekt.fxml"));
                AnchorPane inneresProjekt = (AnchorPane) loader.load();
```

```
                ProjektDetailController detailProjektController = loader
                        .<ProjektDetailController> getController();
                detailProjektController.setRootController(rootController);

                detailProjektController.init(tblProjekt.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresProjekt);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
    @FXML
    private void addProjekt(){
        try {
            // Load ProjektAdd View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AddProjekt.fxml"));
            AnchorPane addProjekt = (AnchorPane) loader.load();

            AddProjektController addProjektController = loader
                    .<AddProjektController> getController();
            addProjektController.setRootController(rootController);

            rootController.rootLayout.setCenter(addProjekt);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

ProjektDetailController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
```

```
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The ProjektDetailController handles all interaction with projects *
 *
 * @author sritz
 * @version 1.0
 *
 */

public class ProjektDetailController implements Initializable {
    //WS Client to interact
    ClientWS client = null;
    RootController rootController = null;
    DateFormat formatDatum = null;
    DateTimeFormatter dateFormatter = null;
    GregorianCalendar timestamp = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Projekt projekt = null;

    //Left Panel
    @FXML
    private Label lblPersonId;
    @FXML
    private Label lblProjektBauherr;
    @FXML
    private TextField txtProjektStrasse;
    @FXML
    private ComboBox<Plz> cbProjektPlz;
    @FXML
    private Label lblProjektOrt;
    @FXML
    private ComboBox<Objekttyp> cbProjektObjekttyp;
    @FXML
    private ComboBox<Arbeitstyp> cbProjektArbeitstyp;
    @FXML
    private Label lblProjektStartdatum;
    @FXML
    private DatePicker dateProjektEnddatum;
```

```
    @FXML
    private Label lblProjektFaellig;
    @FXML
    private ComboBox<Projektstatus> cbProjektStatus;

    //Right Panel Mangel
    @FXML
    private TableView<Mangel> tblProjektMangel;
    @FXML
    private TableColumn<Mangel, String> colProjektMangelId;
    @FXML
    private TableColumn<Mangel, String> colProjektMangelBezeichnung;
    ObservableList<Mangel> mangelData;

    //Right Panel Meldung
    @FXML
    private TableView<Meldung> tblProjektMeldung;
    @FXML
    private TableColumn<Meldung, String> colProjektMeldungId;
    @FXML
    private TableColumn<Meldung, String> colProjektMeldungBezeichnung;

    ObservableList<Meldung> meldungData;

    //Right Panel SubUnternehmen
    @FXML
    private TableView<Subunternehmen> tblProjektUnternehmen;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenId;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenName;
    @FXML
    private TableColumn<Subunternehmen, String>
colProjektUnternehmenStrasse;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenPlz;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenOrt;
    @FXML
    private TableColumn<Subunternehmen, String>
colProjektUnternehmenTelefon;

    ObservableList<Subunternehmen> subunternehmenData;

    @FXML
    private ComboBox<Subunternehmen> cbSubunternehmen;
    @FXML
    private ComboBox<SuMitarbeiter> cbAnsprechperson;



    //Right Panel Bauleiter
    @FXML
    private TableView<ProjektGuMitarbeiter> tblProjektBauleiter;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterId;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterName;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterVorname;
```

```
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterStartdatum;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterEnddatum;

    @FXML
    private ComboBox<GuMitarbeiter> cbProjektBauleiter;



    ObservableList<ProjektGuMitarbeiter> bauleiterData;



    @Override
    public void initialize(URL location, ResourceBundle resources) {
        client = ClientWS.getInstance();
        formatDatum = new SimpleDateFormat("dd.MM.yyyy");
        dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        timestamp = (GregorianCalendar) Calendar.getInstance();

        for (Plz plz :
FXCollections.observableArrayList(client.proxy.getAllPlz())) {
            cbProjektPlz.getItems().add(plz);
        }
        for (Objekttyp objekttyp :
FXCollections.observableArrayList(client.proxy.getAllObjekttyp())) {
            cbProjektObjekttyp.getItems().add(objekttyp);
        }
        for (Arbeitstyp arbeitstyp :
FXCollections.observableArrayList(client.proxy.getAllArbeitstyp())) {
            cbProjektArbeitstyp.getItems().add(arbeitstyp);
        }
        for (Subunternehmen subunternehmen :
FXCollections.observableArrayList(client.proxy.getAllSubunternehmen())) {
            cbSubunternehmen.getItems().add(subunternehmen);
        }
        for (GuMitarbeiter guMitarbeiter :
FXCollections.observableArrayList(client.proxy.getAllGuMitarbeiter())) {
            cbProjektBauleiter.getItems().add(guMitarbeiter);
        }
        for (Projektstatus projektStatus :
FXCollections.observableArrayList(client.proxy.getAllProjektstatus())) {
            cbProjektStatus.getItems().add(projektStatus);
        }


        setCellValueFactoryTblMangel();
        setCellValueFactoryTblMeldung();
        setCellValueFactoryTblUnternehmen();
        setCellValueFactoryTblBauleiter();


    }


    private void setCellValueFactoryTblBauleiter() {
        // TODO Auto-generated method stub
```

```
        colProjektBauleiterId.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                return new
SimpleStringProperty(p.getValue().getFkProjekt().getId().toString());
            }
        });
        colProjektBauleiterName.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                return new
SimpleStringProperty(p.getValue().getFkMitarbeiter().getNachname());
            }
        });
        colProjektBauleiterVorname.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                return new
SimpleStringProperty(p.getValue().getFkMitarbeiter().getVorname());
            }
        });

        colProjektBauleiterStartdatum.setCellValueFactory(new
Callback<CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                if (p.getValue().getStartDatum() == null){
                    return new SimpleStringProperty(" ");
                }else{
                    return new
SimpleStringProperty(formatDatum.format(p.getValue().getStartDatum().getTim
e()));
                }
            }
        });
        colProjektBauleiterEnddatum.setCellValueFactory(new
Callback<CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                if (p.getValue().getEndDatum() == null){
                    return new SimpleStringProperty(" ");
                }else{
                    return new
SimpleStringProperty(formatDatum.format(p.getValue().getEndDatum().getTime(
)));
                }
            }
        });
    }


    private void setCellValueFactoryTblMeldung() {
        colProjektMeldungId.setCellValueFactory(new
PropertyValueFactory<Meldung, String>("id"));
```

```
        colProjektMeldungBezeichnung.setCellValueFactory(new
PropertyValueFactory<Meldung, String>("text"));
    }


    private void setCellValueFactoryTblMangel() {
        colProjektMangelId.setCellValueFactory(new
PropertyValueFactory<Mangel, String>("id"));
        colProjektMangelBezeichnung.setCellValueFactory(new
PropertyValueFactory<Mangel, String>("bezeichnung"));
    }

    private void setCellValueFactoryTblUnternehmen() {
        colProjektUnternehmenId.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("id"));
        colProjektUnternehmenName.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("name"));
        colProjektUnternehmenStrasse.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
                return new
SimpleStringProperty(p.getValue().getFkAdresse().getStrasse());
            }
        });
        colProjektUnternehmenPlz.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
                return new
SimpleStringProperty(p.getValue().getFkAdresse().getPlz().getPlz().toString
());
            }
        });
        colProjektUnternehmenOrt.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
                return new SimpleStringProperty(
p.getValue().getFkAdresse().getPlz().getOrt());
            }
        });
        colProjektUnternehmenTelefon.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("telefon"));
    }


    public void init(int projektId) {
            try {
            projekt = client.proxy.getProjektById(projektId);
            lblPersonId.setText(projekt.getId().toString());

lblProjektBauherr.setText(projekt.getFkBauherr().get(0).getNachname() + " "
+projekt.getFkBauherr().get(0).getVorname());
            txtProjektStrasse.setText(projekt.getFkAdresse().getStrasse());

cbProjektPlz.getSelectionModel().select(projekt.getFkAdresse().getPlz());

cbProjektPlz.setPromptText(projekt.getFkAdresse().getPlz().getPlz().toStrin
g());
```

```
lblProjektOrt.setText(cbProjektPlz.getSelectionModel().getSelectedItem().ge
tOrt());
            cbProjektArbeitstyp.setValue(projekt.getFkArbeitstyp());

cbProjektArbeitstyp.setPromptText(projekt.getFkArbeitstyp().getBezeichnung(
));

cbProjektObjekttyp.getSelectionModel().select(projekt.getFkObjekttyp());

cbProjektObjekttyp.setPromptText(projekt.getFkObjekttyp().getBezeichnung())
;


lblProjektStartdatum.setText(formatDatum.format(projekt.getStartDatum().get
Time()));
            if(projekt.getEndDatum() != null){

dateProjektEnddatum.setValue(LocalDate.parse(formatDatum.format(projekt.get
EndDatum().getTime()), dateFormatter));

            }

lblProjektFaellig.setText(formatDatum.format(projekt.getFaelligkeitsDatum()
.getTime()));

cbProjektStatus.getSelectionModel().select(projekt.getFkProjektstatus());

            mangelData =
FXCollections.observableArrayList(client.proxy.getAllMangelProjekt(projekt.
getId()));
            subunternehmenData =
FXCollections.observableArrayList(client.proxy.getUnternehmenByProjekt(proj
ekt.getId()));
            bauleiterData =
FXCollections.observableArrayList(client.proxy.getBauleiterByProjekt(projek
t));
            tblProjektMangel.setItems(mangelData);
            tblProjektUnternehmen.setItems(subunternehmenData);
            tblProjektBauleiter.setItems(bauleiterData);

        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    @FXML
    public void showMeldungByMangelOderMangel(MouseEvent t) throws
IOException{
        if(t.getClickCount() ==1){
            meldungData =
FXCollections.observableArrayList(client.proxy.getAllMeldungByMangel(tblPro
jektMangel.getSelectionModel().getSelectedItem()));
            tblProjektMeldung.setItems(meldungData);
        }
        else if (t.getClickCount() == 2) {

System.out.println(tblProjektMangel.getSelectionModel().getSelectedItem()
                    .getId());
            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
```

```
                loader.setLocation(Main.class
                        .getResource("view/mangel/InnererMangel.fxml"));
                AnchorPane innererMangel = (AnchorPane) loader.load();

                MangelDetailController mangelDetailController = loader
                        .<MangelDetailController> getController();
                mangelDetailController.setRootController(rootController);


mangelDetailController.init(tblProjektMangel.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(innererMangel);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

    @FXML
    public void projektCancel(){
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AusseresProjekt.fxml"));
            AnchorPane ausseresProjekt = (AnchorPane) loader.load();

            ProjektController projektController =
loader.<ProjektController>getController();
            projektController.setRootController(rootController);

            rootController.rootLayout.setCenter(ausseresProjekt);


        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @FXML
    public void fillCbAnsprechperson(){
        for (SuMitarbeiter sumitarbeiter :
client.proxy.getAllSubunternehmenMitarbeiter(cbSubunternehmen.getSelectionM
odel().getSelectedItem())) {
            cbAnsprechperson.getItems().add(sumitarbeiter);
        }
    }

    @FXML
    public void projektSave(){

        projekt.getFkAdresse().setStrasse(txtProjektStrasse.getText());
        projekt.getFkAdresse().getPlz().setPlz((Integer)
cbProjektPlz.getSelectionModel().getSelectedItem().getPlz());
        projekt.getFkAdresse().getPlz().setOrt(lblProjektOrt.getText());


projekt.setFkObjekttyp(cbProjektObjekttyp.getSelectionModel().getSelectedIt
em());

projekt.setFkArbeitstyp(cbProjektArbeitstyp.getSelectionModel().getSelected
Item());
```

```
        if(dateProjektEnddatum.getValue() != null){
            projekt.setEndDatum(new
GregorianCalendar(dateProjektEnddatum.getValue().getYear(),
dateProjektEnddatum.getValue().getMonthValue() -1,
dateProjektEnddatum.getValue().getDayOfMonth()));
        }

projekt.setFkProjektstatus(cbProjektStatus.getSelectionModel().getSelectedI
tem());
        client.proxy.updateProjekt(projekt);
        try {
            // Load Projekt overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AusseresProjekt.fxml"));
            AnchorPane projekte = (AnchorPane) loader.load();

            ProjektController projektController =
loader.<ProjektController>getController();
            projektController.setRootController(rootController);

            rootController.rootLayout.setCenter(projekte);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @FXML
    public void projektAddMangel(){
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/mangel/AddMangel.fxml"));
            AnchorPane addMangel = (AnchorPane) loader.load();

            AddMangelController addMangelController =
loader.<AddMangelController>getController();
            addMangelController.setRootController(rootController);

            addMangelController.init(projekt);
            rootController.rootLayout.setCenter(addMangel);


        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @FXML
    public void projektAddMeldung(){
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/meldung/AddMeldung.fxml"));
            AnchorPane addMeldung = (AnchorPane) loader.load();

            AddMeldungController addMeldungController =
loader.<AddMeldungController>getController();
            addMeldungController.setRootController(rootController);
```

```
addMeldungController.init(tblProjektMangel.getSelectionModel().getSelectedI
tem());
            rootController.rootLayout.setCenter(addMeldung);


        } catch (IOException e) {
            e.printStackTrace();
        }
    }




    @FXML
    private void showMeldungDetail(MouseEvent t) throws IOException{
        if (t.getClickCount() == 2) {

System.out.println(tblProjektMeldung.getSelectionModel().getSelectedItem()
                    .getId());

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/InnereMeldung.fxml"));
                AnchorPane innereMeldung = (AnchorPane) loader.load();

                MeldungDetailController meldungDetailController = loader
                        .<MeldungDetailController> getController();
                meldungDetailController.setRootController(rootController);


meldungDetailController.init(tblProjektMeldung.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(innereMeldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    private void showSubunternehmenDetail(MouseEvent t) throws IOException{
        if (t.getClickCount() == 2) {

System.out.println(tblProjektUnternehmen.getSelectionModel().getSelectedIte
m()
                    .getId());

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class

.getResource("view/unternehmen/InneresUnternehmen.fxml"));
                AnchorPane inneresUnternehmen = (AnchorPane) loader.load();

                SubUnternehmenDetailController
subunternehmenDetailController = loader
                        .<SubUnternehmenDetailController> getController();
```

```
subunternehmenDetailController.setRootController(rootController);


subunternehmenDetailController.init(tblProjektUnternehmen.getSelectionModel
()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresUnternehmen);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    @FXML
    private void showBauleiterDetail(MouseEvent t) throws IOException{
        if (t.getClickCount() == 2) {

System.out.println(tblProjektBauleiter.getSelectionModel().getSelectedItem(
)
                    .getId());

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/person/InnerePerson.fxml"));
                AnchorPane innerePerson = (AnchorPane) loader.load();

                PersonDetailController personDetailController = loader
                        .<PersonDetailController> getController();
                personDetailController.setRootController(rootController);


personDetailController.init(tblProjektBauleiter.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(innerePerson);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    public void projektAddUnternehmen(){
        client.proxy.addSuMitarbeiterByProjekt(new
ProjektSuMitarbeiter(projekt,
cbAnsprechperson.getSelectionModel().getSelectedItem(),timestamp, null));
        subunternehmenData =
FXCollections.observableArrayList(client.proxy.getUnternehmenByProjekt(proj
ekt.getId()));
        tblProjektUnternehmen.setItems(subunternehmenData);
    }

    @FXML
    public void projektAddBauleiter(){
        //Letzter Bauleiter Enddatum sezten
        ProjektGuMitarbeiter lastBauleiter=
tblProjektBauleiter.getItems().get(tblProjektBauleiter.getItems().size()-
1);
        lastBauleiter.setEndDatum(timestamp);
        client.proxy.updateProjektGuMitarbeiter(lastBauleiter);
```

```
        bauleiterData =
FXCollections.observableArrayList(client.proxy.getBauleiterByProjekt(projek
t));
        tblProjektBauleiter.setItems(bauleiterData);
        client.proxy.addGuMitarbeiterByProjekt(new
ProjektGuMitarbeiter(projekt,
cbProjektBauleiter.getSelectionModel().getSelectedItem(), timestamp,
null));
        bauleiterData =
FXCollections.observableArrayList(client.proxy.getBauleiterByProjekt(projek
t));
        tblProjektBauleiter.setItems(bauleiterData);
    }

    @FXML
    private void plzChange(){
        if (cbProjektPlz.getSelectionModel().getSelectedItem() != null){

lblProjektOrt.setText(cbProjektPlz.getSelectionModel().getSelectedItem().ge
tOrt());
        }
    }

}
```

RootController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import ch.hsluw.mangelmanager.client.extern.Main;

/**
 * The RootController is used to load different content
 * in center of the BorderPane
 *
 *
 * @author sritz, mmonti
 * @version 1.0
 *
 */
public class RootController implements Initializable {


    @FXML
    public BorderPane rootLayout;

    @FXML
    private void logout() {
        // Load Login and close current Stage
        Main.initRootLayout();
        Stage stageToClose = (Stage) rootLayout.getScene().getWindow();
        stageToClose.close();
```

```
    }
    @FXML
    private void showPersonen() {
        try {
            // Load Person overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
            AnchorPane personen = (AnchorPane) loader.load();

            PersonController personController =
loader.<PersonController>getController();
            personController.setRootController(this);

            rootLayout.setCenter(personen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showUnternehmen() {
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
            subunternehmenController.setRootController(this);

            rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showProjekte() {
        try {
            // Load Projekt overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AusseresProjekt.fxml"));
            AnchorPane projekte = (AnchorPane) loader.load();

            ProjektController projektController =
loader.<ProjektController>getController();
            projektController.setRootController(this);

            rootLayout.setCenter(projekte);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
```

```
    private void showMaengel() {
        try {
            // Load Maengel overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/mangel/AussererMangel.fxml"));
            AnchorPane maengel = (AnchorPane) loader.load();

            MangelController mangelController =
loader.<MangelController>getController();
            mangelController.setRootController(this);

            rootLayout.setCenter(maengel);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showMeldungen() {
        try {
            // Load Meldung overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/meldung/AussereMeldung.fxml"));
            AnchorPane meldungen = (AnchorPane) loader.load();

            MeldungController meldungController =
loader.<MeldungController>getController();
            meldungController.setRootController(this);

            rootLayout.setCenter(meldungen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        // TODO Auto-generated method stub
        showPersonen();
    }

}
```

SubUnternehmenController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
```

```
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The SubunternehmenController handles all interaction with subunternehmen
*
 *
 * @author lkuendig
 * @version 1.0
 *
 */

public class SubUnternehmenController implements Initializable {
    //WS Client to interact
    ClientWS client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    //Define overviewtable with columns
    @FXML
    private TableView<Subunternehmen> tblSubunternehmen;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenId;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenName;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenAdresse;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenTelefon;
    @FXML
    private TableColumn<Subunternehmen, String>
colSubunternehmenOffeneProjekte;

    //Datalist for Tableview
    ObservableList<Subunternehmen> data;


    //SetCellValueFactory from overviewtable
    @Override
    public void initialize(URL location, ResourceBundle resources) {

        colSubunternehmenId.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("id"));
        colSubunternehmenName.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("name"));

        colSubunternehmenAdresse.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
```

```
                return new
SimpleStringProperty(p.getValue().getFkAdresse().getStrasse() + " "
+p.getValue().getFkAdresse().getPlz().getPlz() + " " +
p.getValue().getFkAdresse().getPlz().getOrt());
            }
        });
        colSubunternehmenTelefon.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("telefon"));

        colSubunternehmenOffeneProjekte.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
                return new
SimpleStringProperty(client.proxy.getProjektproSubunternehmen(p.getValue().
getId()));
            }
        });


        //Client interaction
        try {
            client = new ClientWS();
            data =
FXCollections.observableArrayList(client.proxy.getAllSubunternehmen());
        } catch (Exception e) {
            e.printStackTrace();
        }


        //Set data to tableview
        tblSubunternehmen.setItems(data);
    }

    @FXML
    public void showSubunternehmenDetail(MouseEvent t) throws IOException{
        if(t.getClickCount() == 2){

System.out.println(tblSubunternehmen.getSelectionModel().getSelectedItem().
getId());

            try {
                // Load SubunternehmenDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class

.getResource("view/unternehmen/InneresUnternehmen.fxml"));
                AnchorPane inneresUnternehmen = (AnchorPane) loader.load();

                SubUnternehmenDetailController
detailSubunternehmenController =
loader.<SubUnternehmenDetailController>getController();

detailSubunternehmenController.setRootController(rootController);


detailSubunternehmenController.init(tblSubunternehmen.getSelectionModel().g
etSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresUnternehmen);


            } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }
    }
    @FXML
    private void addUnternehmen(){
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/unternehmen/AddUnternehmen.fxml"));
            AnchorPane addUnternehmen = (AnchorPane) loader.load();

            AddUnternehmenController addUnternehmenController =
loader.<AddUnternehmenController>getController();
            addUnternehmenController.setRootController(rootController);
            rootController.rootLayout.setCenter(addUnternehmen);


        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

SubUnternehmenDetailController

```
package ch.hsluw.mangelmanager.client.extern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.extern.ClientWS;
import ch.hsluw.mangelmanager.client.extern.Main;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
```

```
public class SubUnternehmenDetailController implements Initializable {

    //WS Client to interact
        ClientWS client = null;
        RootController rootController = null;
        DateFormat formatDatum = null;
        DateTimeFormatter dateFormatter = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        Subunternehmen subunternehmen = null;

        @FXML
        public Label lblUnternehmenId;
        @FXML
        public TextField txtUnternehmenName;
        @FXML
        public TextField txtUnternehmenTelefon;
        @FXML
        public TextField txtUnternehmenStrasse;
        @FXML
        public ComboBox<Plz> cbUnternehmenPlz;
        @FXML
        public Label lblUnternehmenOrt;

        //Projekte pro Subunternehmen
        @FXML
        private TableView<Projekt> tblUnternehmenProjekt;
        @FXML
        private TableColumn<Projekt, String> colUnternehmenProjektId;
        @FXML
        private TableColumn<Projekt, String>
colUnternehmenProjektBezeichnung;
        @FXML
        private TableColumn<Projekt, String> colUnternehmenProjektBauherr;
        @FXML
        private TableColumn<Projekt, String> colUnternehmenProjektStrasse;
        @FXML
        private TableColumn<Projekt, String> colUnternehmenProjektPlz;
        @FXML
        private TableColumn<Projekt, String> colUnternehmenProjektOrt;
        @FXML
        private TableColumn<Projekt, String>
colUnternehmenProjektStartdatum;
        @FXML
        private TableColumn<Projekt, String> colUnternehmenProjektStatus;

        //Projekte pro Subunternehmen
                @FXML
                private TableView<SuMitarbeiter> tblUnternehmenMitarbeiter;
                @FXML
                private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterId;
                @FXML
                private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterName;
                @FXML
                private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterVorname;
```

```
                @FXML
                private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterStartdatum;
                @FXML
                private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterEnddatum;
                @FXML
                private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterTelefon;


        //Datalist for Tableview
        ObservableList<Projekt> data;
        ObservableList<SuMitarbeiter>data2;

        @Override
        public void initialize(URL location, ResourceBundle resources) {


            try {
                client = ClientWS.getInstance();
                for (Plz plz :
FXCollections.observableArrayList(client.proxy.getAllPlz())) {
                    cbUnternehmenPlz.getItems().add(plz);
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }




        public void init(int subunternehmenId) {
            setCellValueFactoryTblUnternehmenProjekt();
            setCellValueFactoryTblUnternehmenMiterbeiter();
                try {

                formatDatum = new SimpleDateFormat("dd.MM.yyyy");
                dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
                subunternehmen =
client.proxy.getSubunternehmenById(subunternehmenId);

lblUnternehmenId.setText(subunternehmen.getId().toString());
                txtUnternehmenName.setText(subunternehmen.getName());
                txtUnternehmenTelefon.setText(subunternehmen.getTelefon());

txtUnternehmenStrasse.setText(subunternehmen.getFkAdresse().getStrasse());

cbUnternehmenPlz.setValue(subunternehmen.getFkAdresse().getPlz());

lblUnternehmenOrt.setText(subunternehmen.getFkAdresse().getPlz().getOrt());

                data =
FXCollections.observableArrayList(client.proxy.getAllSubunternehmenProjekt(
subunternehmen.getId()));
                data2 =
FXCollections.observableArrayList(client.proxy.getAllSubunternehmenMitarbei
ter(subunternehmen));
                //Set data to tableview
                tblUnternehmenProjekt.setItems(data);
```

```
                tblUnternehmenMitarbeiter.setItems(data2);
            } catch (Exception e) {
                e.printStackTrace();
            }

        }

        private void setCellValueFactoryTblUnternehmenProjekt() {
            // TODO Auto-generated method stub
            colUnternehmenProjektId.setCellValueFactory(new
PropertyValueFactory<Projekt, String>("id"));
            colUnternehmenProjektBezeichnung.setCellValueFactory(new
PropertyValueFactory<Projekt, String>("bezeichnung"));

            colUnternehmenProjektBauherr.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
                public ObservableValue<String> call(
                        CellDataFeatures<Projekt, String> p) {
                    return new SimpleStringProperty(p.getValue()
                            .getFkBauherr().get(0).getNachname()
                            + " "
                            + p.getValue().getFkBauherr().get(0)
                                    .getVorname());
                }
            });

            colUnternehmenProjektStrasse.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
                public ObservableValue<String> call(
                        CellDataFeatures<Projekt, String> p) {return new
SimpleStringProperty(p.getValue().getFkAdresse().getStrasse());}
            });
            colUnternehmenProjektPlz.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
                public ObservableValue<String> call(
                CellDataFeatures<Projekt, String> p) {return new
SimpleStringProperty(p.getValue().getFkAdresse().getPlz().getPlz().toString
());}
            });
            colUnternehmenProjektOrt.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
                public ObservableValue<String> call(
                CellDataFeatures<Projekt, String> p) {return new
SimpleStringProperty(p.getValue().getFkAdresse().getPlz().getOrt());}
            });

            colUnternehmenProjektStartdatum.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
                public ObservableValue<String> call(
                        CellDataFeatures<Projekt, String> p) {return new
SimpleStringProperty(formatDatum.format(p.getValue().getStartDatum().getTim
e()));}
            });

            colUnternehmenProjektStatus.setCellValueFactory(new
Callback<CellDataFeatures<Projekt, String>, ObservableValue<String>>() {
                public ObservableValue<String> call(
                CellDataFeatures<Projekt, String> p) {return new
SimpleStringProperty(p.getValue().getFkProjektstatus().getBezeichnung());}
            });
        }

        public void setCellValueFactoryTblUnternehmenMiterbeiter(){
```

```
            colUnternehmenMitarbeiterName.setCellValueFactory(new
PropertyValueFactory<SuMitarbeiter, String>("nachname"));
            colUnternehmenMitarbeiterVorname.setCellValueFactory(new
PropertyValueFactory<SuMitarbeiter, String>("vorname"));
            colUnternehmenMitarbeiterTelefon.setCellValueFactory(new
PropertyValueFactory<SuMitarbeiter, String>("telefon"));
        }

        @FXML
        private void unternehmenSave() {

            subunternehmen.setName(txtUnternehmenName.getText());
            subunternehmen.setTelefon(txtUnternehmenTelefon.getText());

subunternehmen.getFkAdresse().setPlz(cbUnternehmenPlz.getSelectionModel().g
etSelectedItem());

subunternehmen.getFkAdresse().getPlz().setOrt(lblUnternehmenOrt.getText());

subunternehmen.getFkAdresse().setStrasse(txtUnternehmenStrasse.getText());
            client.proxy.updateSubunternehmen(subunternehmen);

            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
                AnchorPane unternehmen = (AnchorPane) loader.load();

                SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
                subunternehmenController.setRootController(rootController);

                rootController.rootLayout.setCenter(unternehmen);

            } catch (IOException e) {
                e.printStackTrace();
            }

        }

        @FXML
        private void unternehmenCancel(){
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
                AnchorPane unternehmen = (AnchorPane) loader.load();

                SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
                subunternehmenController.setRootController(rootController);

                rootController.rootLayout.setCenter(unternehmen);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        @FXML
```

```
        private void plzChange(){
            if (cbUnternehmenPlz.getSelectionModel().getSelectedItem() !=
null){

lblUnternehmenOrt.setText(cbUnternehmenPlz.getSelectionModel().getSelectedI
tem().getOrt());
            }else{
            }
        }

        @FXML
        private void showProjektDetail(MouseEvent t) throws IOException{
            if (t.getClickCount() == 2) {

System.out.println(tblUnternehmenProjekt.getSelectionModel().getSelectedIte
m()
                        .getId());

                try {
                    // Load ProjektDetail View.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class

.getResource("view/projekt/InneresProjekt.fxml"));
                    AnchorPane inneresProjekt = (AnchorPane) loader.load();

                    ProjektDetailController detailProjektController =
loader
                            .<ProjektDetailController> getController();

detailProjektController.setRootController(rootController);


detailProjektController.init(tblUnternehmenProjekt.getSelectionModel()
                            .getSelectedItem().getId());
                    rootController.rootLayout.setCenter(inneresProjekt);

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        @FXML
        public void showMitarbeiterDetail(MouseEvent t){
            if (t.getClickCount() == 2) {

System.out.println(tblUnternehmenMitarbeiter.getSelectionModel().getSelecte
dItem()
                        .getId());

                try {
                    // Load ProjektDetail View.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class
                            .getResource("view/person/InnerePerson.fxml"));
                    AnchorPane inneresPerson = (AnchorPane) loader.load();

                    PersonDetailController detailPersonController = loader
                            .<PersonDetailController> getController();

detailPersonController.setRootController(rootController);
```

```
detailPersonController.init(tblUnternehmenMitarbeiter.getSelectionModel().g
etSelectedItem()
                                .getId());
                        rootController.rootLayout.setCenter(inneresPerson);

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }

}
```

ClientRMI

```
package ch.hsluw.mangelmanager.client.intern;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.util.List;

import org.apache.log4j.Logger;

import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
import ch.hsluw.mangelmanager.rmi.adresse.AdresseRO;
import ch.hsluw.mangelmanager.rmi.arbeitstyp.ArbeitstypRO;
import ch.hsluw.mangelmanager.rmi.bauherr.BauherrRO;
import ch.hsluw.mangelmanager.rmi.gumitarbeiter.GuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.login.LoginRO;
import ch.hsluw.mangelmanager.rmi.mangel.MangelRO;
import ch.hsluw.mangelmanager.rmi.mangelstatus.MangelstatusRO;
import ch.hsluw.mangelmanager.rmi.meldung.MeldungRO;
import ch.hsluw.mangelmanager.rmi.meldungstyp.MeldungstypRO;
import ch.hsluw.mangelmanager.rmi.objekttyp.ObjekttypRO;
import ch.hsluw.mangelmanager.rmi.person.PersonRO;
import ch.hsluw.mangelmanager.rmi.plz.PlzRO;
import ch.hsluw.mangelmanager.rmi.projekt.ProjektRO;
import
ch.hsluw.mangelmanager.rmi.projektgumitarbeiter.ProjektGuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.projektstatus.ProjektstatusRO;
import
ch.hsluw.mangelmanager.rmi.projektsumitarbeiter.ProjektSuMitarbeiterRO;
```

```java
import ch.hsluw.mangelmanager.rmi.rolle.RolleRO;
import ch.hsluw.mangelmanager.rmi.subunternehmen.SubunternehmenRO;
import ch.hsluw.mangelmanager.rmi.sumitarbeiter.SuMitarbeiterRO;



/**
 * Diese Klasse stellt das Userinterface fuer die Modulverwaltung via RMI
zur
 * Verfuegung
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ClientRMI {
    private static ClientRMI instance;

    public static ClientRMI getInstance () {
        if (ClientRMI.instance == null) {
            try {
                ClientRMI.instance = new ClientRMI();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        return ClientRMI.instance;
      }

    List<Person> person;
    List<Projekt> projekte;
    List<Projekt> suprojekte;
    List<Projektstatus> projektstatus;
    List<Subunternehmen> subunternehmen;

    List<Mangel> maengel;
    List<Meldung> meldung;
    List<Plz> plz;
    List<Objekttyp> objekttyp;
    List<Arbeitstyp> arbeitstyp;
    List<SuMitarbeiter> sumitarbeiter;
    List<ProjektGuMitarbeiter> bauleiter;
    List<Mangel> mangelOfProjekt;
    List<Meldung> meldungByMangel;
    List<Bauherr> bauherren;
    List<GuMitarbeiter> guMitarbeiter;

    List<Mangelstatus> mangelstatus;
    List<Meldungstyp> meldungstyp;
    String anzProjekte;
    Projekt projekt;
    Subunternehmen subunternehmennr;
    Meldung meldungnr;
    Plz plznr;
    Mangel mangelnr;
    Adresse addAdresse;
    Login login;
    Login loginnr;
    Person personnr;
    List<Rolle> rollen;
    String url;
```

```
private static Logger logger = Logger.getLogger(ClientRMI.class);
PersonRO personRO;
ProjektRO projektRO;
SubunternehmenRO subunternehmenRO;
MangelRO mangelRO;
MeldungRO meldungRO;
PlzRO plzRO;
AdresseRO adresseRO;
ObjekttypRO  objekttypRO;
ArbeitstypRO arbeitstypRO;
MangelstatusRO mangelstatusRO;
MeldungstypRO meldungstypRO;
ProjektGuMitarbeiterRO projektGuMitarbeiterRO;
LoginRO loginRO;
ProjektSuMitarbeiterRO projektSuMitarbeiterRO;
BauherrRO bauherrRO;
GuMitarbeiterRO guMitarbeiterRO;
ProjektstatusRO projektstatusRO;
SuMitarbeiterRO suMitarbeiterRO;
RolleRO rolleRO;

public static void main(String[] args) {
    try {
        // Init Application over RMI
        ClientRMI rmicon = new ClientRMI();
        System.out.println("Verbindung zu RMI Server hergestellt");
    } catch (Exception e) {
        logger.error("RMI Fehler: ", e);
        e.printStackTrace();

    }
}

/**
 * Instantiates a new modulverwaltung client rmi.
 *
 * @throws Exception
 */
public ClientRMI() throws Exception {


    // init rmi connection
    if(Main.ip == null | Main.port == null){
        url = "rmi://localhost:1099/";
    }else{
        url = "rmi://"+Main.ip+":"+Main.port+"/";
    }
    String personROName = "personRO";
    String projektROName = "projektRO";
    String subunternehmenROName = "subunternehmenRO";
    String mangelROName = "mangelRO";
    String meldungROName ="meldungRO";
    String plzROName ="plzRO";
    String adresseROName = "adresseRO";
    String objekttypROName = "objekttypRO";
    String arbeitstyROName = "arbeitstypRO";
    String mangelstatusROName = "mangelstatusRO";
    String meldungstypROName = "meldungstypRO";
    String projektGuMitarbeiterROName = "projektGuMitarbeiterRO";
```

```
        String loginROName = "loginRO";
        String projektSuMitarbeiterROName = "projektSuMitarbeiterRO";
        String bauherrROName = "bauherrRO";
        String guMitarbeiterROName = "guMitarbeiterRO";
        String suMitarbeiterROName = "suMitarbeiterRO";
        String projektstatusROName = "projektstatusRO";
        String rolleROName = "rolleRO";

        this.personRO = (PersonRO) Naming.lookup(url + personROName);
        this.projektRO = (ProjektRO) Naming.lookup(url + projektROName);
        this.mangelRO = (MangelRO) Naming.lookup(url + mangelROName);
        this.meldungRO = (MeldungRO) Naming.lookup(url + meldungROName);
        this.subunternehmenRO = (SubunternehmenRO) Naming.lookup(url +
subunternehmenROName);
        this.plzRO = (PlzRO) Naming.lookup(url + plzROName);
        this.adresseRO = (AdresseRO) Naming.lookup(url + adresseROName);
        this.objekttypRO = (ObjekttypRO) Naming.lookup(url +
objekttypROName);
        this.arbeitstypRO = (ArbeitstypRO) Naming.lookup(url +
arbeitstyROName);
        this.mangelstatusRO = (MangelstatusRO) Naming.lookup(url +
mangelstatusROName);
        this.meldungstypRO = (MeldungstypRO) Naming.lookup(url +
meldungstypROName);
        this.projektGuMitarbeiterRO = (ProjektGuMitarbeiterRO)
Naming.lookup(url + projektGuMitarbeiterROName);
        this.loginRO = (LoginRO) Naming.lookup(url+ loginROName);
        this.projektSuMitarbeiterRO = (ProjektSuMitarbeiterRO)
Naming.lookup(url + projektSuMitarbeiterROName);
        this.bauherrRO = (BauherrRO) Naming.lookup(url + bauherrROName);
        this.guMitarbeiterRO = (GuMitarbeiterRO) Naming.lookup(url +
guMitarbeiterROName);
        this.projektstatusRO = (ProjektstatusRO) Naming.lookup(url +
projektstatusROName);
        this.suMitarbeiterRO = (SuMitarbeiterRO) Naming.lookup(url +
suMitarbeiterROName);
        this.rolleRO = (RolleRO) Naming.lookup(url + rolleROName);



    }

    public List<Projekt> getAllProjekt() {
        // TODO Auto-generated method stub
        try {
            projekte = projektRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;

    }

    public List<Subunternehmen> getAllSubunternehmen() {

        try {
            subunternehmen = subunternehmenRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return subunternehmen;
```

```
}

public String getProjektproSubunternehmen(int subunternehmen){
    try {
        anzProjekte = subunternehmenRO.findAllProjekte(subunternehmen);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return anzProjekte;

}


public List<Mangel> getAllMangel() {

    // TODO Auto-generated method stub
    try {
        maengel = mangelRO.findAll();
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return maengel;

}


public List<Meldung> getAllMeldung(){
    try {
        meldung = meldungRO.findAll();
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return meldung;
}

public Projekt getProjektById(int projektId) {
    // TODO Auto-generated method stub
    try {
        projekt = projektRO.findById(projektId);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return projekt;
}

public Subunternehmen getSubunternehmenById(int subunternehmenId) {
    try {
        subunternehmennr = subunternehmenRO.findById(subunternehmenId);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return subunternehmennr;
}

public Meldung getMeldungById(int meldungId) {
    try {
        meldungnr = meldungRO.findById(meldungId);
    } catch (RemoteException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return meldungnr;
    }

    public List<Person> getAllPerson() {
        // TODO Auto-generated method stub
        try {
            person = personRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return person;
    }


    public void updateSubunternehmen(Subunternehmen subunternehmen) {
            try {
                subunternehmenRO.update(subunternehmen);
            } catch (RemoteException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }


    public List<Projekt> getProjektByBezeichnung(String bezeichnung) {
            // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByBezeichnung(bezeichnung);

        }
        catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Plz> getAllPlz() {
        try {
            plz = plzRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return plz;

    }

    public List<Projekt> getProjektByBauherr(String bauherr) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByBauherr(bauherr);

        } catch (RemoteException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByPlz(String plz) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByPlz(plz);

        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByOrt(String ort) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByOrt(ort);

        }
        catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public Plz getPlzById(int plzId) {
        try {
            plznr = plzRO.findById(plzId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return plznr;
    }

    public List<Projekt> getProjektByObjekttyp(String objekttyp) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByObjekttyp(objekttyp);

        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByArbeitstyp(String arbeitstyp) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByArbeitstyp(arbeitstyp);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }
```

```
    public List<Projekt> getProjektByProjektstatus(String projektstatus) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByProjektstatus(projektstatus);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }


    public void addAdresse(Adresse adresse) {
        try {
            adresseRO.add(adresse);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void addSubunternehmen(Subunternehmen addSubunternehmen) {
        try {
            subunternehmenRO.add(addSubunternehmen);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public List<Projekt> getAllSubunternehmenProjekt(Integer
subunternehmen) {
        try {
            suprojekte =
projektRO.findAllSubunternehmenProjekt(subunternehmen);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return suprojekte;
    }

    public List<SuMitarbeiter>
getAllSubunternehmenMitarbeiter(Subunternehmen subunternehmen) {
        try {
            sumitarbeiter =
subunternehmenRO.findAllSubunternehmenMitarbeiter(subunternehmen);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return sumitarbeiter;
    }

    public List<Objekttyp> getAllObjekttyp() {
        // TODO Auto-generated method stub
        try {
            objekttyp = objekttypRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
    return objekttyp;
}

public List<Arbeitstyp> getAllArbeitstyp() {
    // TODO Auto-generated method stub
    try {
        arbeitstyp = arbeitstypRO.findAll();
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return arbeitstyp;
}
public List<Mangel> getAllMangelProjekt(Integer projekt) {
    try {
        mangelOfProjekt = mangelRO.findAllMangelProjekt(projekt);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
        return mangelOfProjekt;
}

public Mangel getMangelById(Integer mangelId) {
    try {
        mangelnr = mangelRO.findById(mangelId);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return mangelnr;
}

public void updateMangel(Mangel mangel) {
    // TODO Auto-generated method stub
    try {
        mangelRO.update(mangel);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public List<Mangelstatus> getAllMangelStatus() {
    try {
        mangelstatus = mangelstatusRO.findAllMangelStatus();

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
        return mangelstatus;
}

public void addMangel(Mangel mangel) {
    // TODO Auto-generated method stub
    try {
        mangelRO.add(mangel);
    } catch (Exception e) {
        // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
    }

    public List<Meldungstyp> getAllMeldungstyp() {
        // TODO Auto-generated method stub
        try {
            meldungstyp = meldungstypRO.findAll();

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return meldungstyp;
    }

    public void addMeldung(Meldung meldung) {
        // TODO Auto-generated method stub
        try {
            meldungRO.add(meldung);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Meldung> getAllMeldungByMangel(Mangel mangel) {
        // TODO Auto-generated method stub
        try {
            meldungByMangel = meldungRO.findAllMeldungByMangel(mangel);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return meldungByMangel;
    }

    public List<Subunternehmen> getUnternehmenByProjekt(Integer projekt2) {
        // TODO Auto-generated method stub
        try {
            subunternehmen =
subunternehmenRO.findAllSubunternehmenByProjekt(projekt2);
        } catch (Exception e) {
            // TODO Auto-generated catch block<
            e.printStackTrace();
        }
            return subunternehmen;
    }

    public List<ProjektGuMitarbeiter> getBauleiterByProjekt(Projekt
projekt2) {
        // TODO Auto-generated method stub
        try {
            bauleiter =
projektGuMitarbeiterRO.findAllBauleiterByProjekt(projekt2);
        } catch (Exception e) {
            e.printStackTrace();
        }
            return bauleiter;
    }

    public Login getLoginByName(String name) {
        // TODO Auto-generated method stub
```

```
        try {
            login = loginRO.findByName(name);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return login;
    }

    public Login getLoginById(Integer loginId) {
        // TODO Auto-generated method stub
        try {
            loginnr = loginRO.findById(loginId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return loginnr;
    }

    public void updateProjekt(Projekt projekt2) {
        // TODO Auto-generated method stub
        try {
            projektRO.update(projekt2);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void addSuMitarbeiterByProjekt(ProjektSuMitarbeiter psum) {
        // TODO Auto-generated method stub
        try {
            projektSuMitarbeiterRO.add(psum);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Bauherr> getAllBauherr() {
        // TODO Auto-generated method stub
        try {
            bauherren = bauherrRO.findAll();

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return bauherren;
    }

    public void addProjekt(Projekt projekt2) {
        // TODO Auto-generated method stub
        try {
            projektRO.add(projekt2);
        } catch (RemoteException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<GuMitarbeiter> getAllGuMitarbeiter() {
        // TODO Auto-generated method stub
        try {
            guMitarbeiter = guMitarbeiterRO.findAll();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return guMitarbeiter;
    }

    public void addGuMitarbeiterByProjekt(
            ProjektGuMitarbeiter projektGuMitarbeiter) {
        // TODO Auto-generated method stub
        try {
            projektGuMitarbeiterRO.add(projektGuMitarbeiter);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public void updateProjektGuMitarbeiter(ProjektGuMitarbeiter
lastBauleiter) {
        // TODO Auto-generated method stub
        try {
            projektGuMitarbeiterRO.update(lastBauleiter);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Projektstatus> getAllProjektstatus() {
        // TODO Auto-generated method stub
        try {
            projektstatus = projektstatusRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return projektstatus;
    }

    public void addGuMitarbeiter(GuMitarbeiter guMitarbeiter) {
        // TODO Auto-generated method stub
        try {
```

```
            guMitarbeiterRO.add(guMitarbeiter);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void addSuMitarbeiter(SuMitarbeiter suMitarbeiter) {
        // TODO Auto-generated method stub
        try {
            suMitarbeiterRO.add(suMitarbeiter);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void addBauherr(Bauherr bauherr) {
        // TODO Auto-generated method stub
        try {
            bauherrRO.add(bauherr);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Rolle> getAllRolle() {
        try {
            rollen = rolleRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return rollen;
    }

    public Person getPersonById(int personId) {
        // TODO Auto-generated method stub
        try {
            personnr = personRO.findById(personId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return personnr;
    }

    public List<Projekt> getProjektbyPerson(Person person) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findProjektbyPerson(person);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public void updatePerson(Person person) {
        // TODO Auto-generated method stub
        try {
            personRO.update(person);
        } catch (Exception e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
    }

    public void updateMeldung(Meldung meldung2) {
        // TODO Auto-generated method stub
        try {
            meldungRO.update(meldung2);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }



}
```

Main

```
package ch.hsluw.mangelmanager.client.intern;


import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import ch.hsluw.mangelmanager.client.intern.controller.MeldungController;
import ch.hsluw.mangelmanager.client.intern.controller.RootController;
import ch.hsluw.mangelmanager.model.Login;

public class Main extends Application {

    // RMI Client to interact
        ClientRMI client = null;

    private static Stage loginStage;
    private static AnchorPane loginLayout;
    public static Integer loginId;
    public static String port;
    public static String ip;

    @FXML
    private Label lblLoginError;
    @FXML
    private TextField txtBenutzer;
    @FXML
    private PasswordField pwPasswort;
    @FXML
    private TextField txtIp;
    @FXML
```

```
    private TextField txtPort;

    @Override
    public void start(Stage primaryStage) {
        this.loginStage = primaryStage;
        this.loginStage.setTitle("Mängelmanager");
        initRootLayout();

    }

    /**
     * Initializes the demo  layout.
     */
    public static void initRootLayout() {
        try {
            // Load root layout from fxml file.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/login/Login.fxml"));
            loginLayout = (AnchorPane) loader.load();

            // Show the scene containing the root layout.
            Scene scene = new Scene(loginLayout);
            loginStage.setScene(scene);
            loginStage.show();
            loginStage.setResizable(false);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void login() throws IOException {

        ip = txtIp.getText();
        port = txtPort.getText();

        client = ClientRMI.getInstance();
        Login data = client.getLoginByName(txtBenutzer.getText());


        if (data != null){
            if ((txtBenutzer.getText().equals(data.getBenutzername()))
                    && (pwPasswort.getText().equals(data.getPasswort()))) {
                lblLoginError.setText("Login erfolgreich!");
                loginId = data.getId();
                Stage stage = new Stage();


                Parent root = FXMLLoader.load(getClass().getResource(
                        "view/root/Root.fxml"));




                Scene scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("Mängelmanager");
                stage.setMaximized(true);
                stage.show();
                Stage stageToClose = (Stage)
txtBenutzer.getScene().getWindow();
                stageToClose.close();
```

```
            } else {
                lblLoginError.setText("Fehler beim Einloggen!");
            }
        } else {
            lblLoginError.setText("Fehler beim Einloggen!");
        }

    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

AddMangelController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.DatePicker;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Projekt;

public class AddMangelController implements Initializable {
        //RMI Client to interact
        ClientRMI client = null;
        RootController rootController = null;
        Projekt projekt = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        Meldung meldung = null;
        Mangel mangel = null;
        Mangelstatus mangelstatus = null;
        Login login = null;
        List<Mangelstatus> mangelstatusl = null;
        List<Meldungstyp> meldungstypl = null;
        Meldungstyp meldungstyp = null;


        @FXML
        public TextField txtMangelBezeichung;
```

```
        @FXML
        public TextArea txtMangelBeschreibung;
        @FXML
        public DatePicker dateMangelFaellig;


        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            // TODO Auto-generated method stub

        }

        public void init(Projekt projekt) {
            try {
                client = new ClientRMI();

                this.projekt = projekt;
                mangelstatusl = client.getAllMangelStatus();
                for (Mangelstatus mangelstatus : mangelstatusl) {
                    if(mangelstatus.getBezeichnung().equals("Offen")){
                        this.mangelstatus = mangelstatus;
                    }
                }
                meldungstypl = client.getAllMeldungstyp();
                for (Meldungstyp meldungstyp : meldungstypl) {
                    if(meldungstyp.getBezeichnung().equals("Reklamation")){
                        this.meldungstyp = meldungstyp;
                    }
                }
                login = client.getLoginById(Main.loginId);

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        @FXML
        private void mangelSave() {
            Calendar cl = Calendar.getInstance();

            mangel = new Mangel(projekt,
txtMangelBezeichung.getText(),(GregorianCalendar) cl, new
GregorianCalendar(dateMangelFaellig.getValue().getYear(),
dateMangelFaellig.getValue().getMonthValue()-1,
dateMangelFaellig.getValue().getDayOfMonth()),mangelstatus ,login ,
txtMangelBeschreibung.getText());

            client.addMangel(mangel);

            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/mangel/AussererMangel.fxml"));
                AnchorPane mangel = (AnchorPane) loader.load();

                MangelController mangelController =
loader.<MangelController>getController();
                mangelController.setRootController(rootController);

                rootController.rootLayout.setCenter(mangel);
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        @FXML
        public void addMangelCancel(){
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/mangel/AussererMangel.fxml"));
                AnchorPane mangel = (AnchorPane) loader.load();

                MangelController mangelController =
loader.<MangelController>getController();
                mangelController.setRootController(rootController);

                rootController.rootLayout.setCenter(mangel);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

AddMeldungController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;

public class AddMeldungController implements Initializable {
        //RMI Client to interact
        ClientRMI client = null;
        RootController rootController = null;
        Mangel mangel = null;
        GregorianCalendar timestamp = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
```

```
        }

        Login login = null;


        @FXML
        public Label lblMeldungProjekt;
        @FXML
        private Label lblMeldungMangel;
        @FXML
        private ChoiceBox<Meldungstyp> cbMeldungstyp;
        @FXML
        private TextArea txtMeldungBeschreibung;


        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            // TODO Auto-generated method stub
            try {
                client = new ClientRMI();
                for (Meldungstyp meldungstyp : client.getAllMeldungstyp())
{
                    cbMeldungstyp.getItems().add(meldungstyp);
                }



            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        public void init(Mangel mangel) {
            try {
                this.mangel = mangel;

lblMeldungProjekt.setText(mangel.getFkProjekt().getBezeichnung());
                lblMeldungMangel.setText(mangel.getBezeichnung());
                login = client.getLoginById(Main.loginId);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        @FXML
        public void meldungSave() {
            timestamp = (GregorianCalendar) Calendar.getInstance();
            client.addMeldung(new Meldung(mangel,
cbMeldungstyp.getSelectionModel().getSelectedItem(),
txtMeldungBeschreibung.getText(),timestamp, false, login));

            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
```

```
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        @FXML
        public void addMeldungCancel(){
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

AddPersonController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.TitledPane;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The AddPersonController handles all interaction with person *
```

```
 *
 * @author sritz
 * @version 1.0
 *
 */

public class AddPersonController implements Initializable {
    //RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Bauherr bauherr;
    GuMitarbeiter guMitarbeiter;
    SuMitarbeiter suMitarbeiter;
    Login login;
    Adresse adresse;

    //Declare FXML objects
    @FXML
    private TextField txtPersonName;
    @FXML
    private TextField txtPersonVorname;
    @FXML
    private TextField txtPersonStrasse;
    @FXML
    private TextField txtPersonTelefon;
    @FXML
    private Label lblPersonOrt;
    @FXML
    private ComboBox<Plz> cbPersonPlz;
    @FXML
    private ComboBox cbPersonFunktion;
    @FXML
    private ChoiceBox<Subunternehmen> cbPersonUnternehmen;
    @FXML
    private Label lblPersonError;
    @FXML
    private TitledPane tpPersonLogin;
    @FXML
    private TextField txtPersonBenutzername;
    @FXML
    private TextField txtPersonPasswort;
    @FXML
    private TextField txtPersonPasswortWiederholen;
    @FXML
    private TextField txtPersonEmail;
    @FXML
    private ComboBox<Rolle> cbPersonRolle;

    //Saves a new Person objekt
    @FXML
    public void personSave(){
        // TODO
        if(cbPersonFunktion.getValue() != null){
            if(cbPersonFunktion.getValue().equals("GuMitarbeiter")){

if(txtPersonPasswort.getText().equals(txtPersonPasswortWiederholen.getText(
))){
```

```
                    login = new Login(txtPersonBenutzername.getText(),
txtPersonPasswort.getText(), txtPersonEmail.getText(),
cbPersonRolle.getSelectionModel().getSelectedItem());
                    guMitarbeiter = new
GuMitarbeiter(txtPersonName.getText(), txtPersonVorname.getText(),
txtPersonTelefon.getText(), login);
                    client.addGuMitarbeiter(guMitarbeiter);
                }else{
                    lblPersonError.setText("Passwort wiederholen!");
                    return;
                }
            }else if(cbPersonFunktion.getValue().equals("SuMitarbeiter")){

if(txtPersonPasswort.getText().equals(txtPersonPasswortWiederholen.getText(
))){
                    login = new Login(txtPersonBenutzername.getText(),
txtPersonPasswort.getText(), txtPersonEmail.getText(),
cbPersonRolle.getSelectionModel().getSelectedItem());
                    suMitarbeiter = new
SuMitarbeiter(txtPersonName.getText(), txtPersonVorname.getText(),
txtPersonTelefon.getText(),
cbPersonUnternehmen.getSelectionModel().getSelectedItem(), login);
                    client.addSuMitarbeiter(suMitarbeiter);
                }else{
                    lblPersonError.setText("Passwort wiederholen!");
                    return;
                }
            }else{
                adresse = new Adresse(txtPersonStrasse.getText(),
cbPersonPlz.getSelectionModel().getSelectedItem());
                bauherr = new Bauherr(txtPersonName.getText(),
txtPersonVorname.getText(), txtPersonTelefon.getText(), adresse);
                client.addBauherr(bauherr);
            }
        }else{
            lblPersonError.setText("Funktion Auswählen");
            return;
        }
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
            AnchorPane person = (AnchorPane) loader.load();

            PersonController personController =
loader.<PersonController>getController();
            personController.setRootController(rootController);

            rootController.rootLayout.setCenter(person);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void personCancel(){
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
```

```
            AnchorPane person = (AnchorPane) loader.load();

            PersonController personController =
loader.<PersonController>getController();
            personController.setRootController(rootController);

            rootController.rootLayout.setCenter(person);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //enabled or disabled fields if not needed
    @FXML
    public void enableFields(){
        if(cbPersonFunktion.getValue() != null){
            if(cbPersonFunktion.getValue().equals("GuMitarbeiter")){
                tpPersonLogin.setDisable(false);
                cbPersonUnternehmen.setDisable(true);
                cbPersonPlz.setDisable(true);
                txtPersonStrasse.setDisable(true);

            }
            else if(cbPersonFunktion.getValue().equals("SuMitarbeiter")){
                tpPersonLogin.setDisable(false);
                cbPersonUnternehmen.setDisable(false);
                cbPersonPlz.setDisable(true);
                txtPersonStrasse.setDisable(true);
            }
            else if(cbPersonFunktion.getValue().equals("Bauherr")){
                tpPersonLogin.setDisable(true);
                cbPersonUnternehmen.setDisable(true);
                cbPersonPlz.setDisable(false);
                txtPersonStrasse.setDisable(false);
            }
        }
    }



    @FXML
    private void plzChange(){
        if (cbPersonPlz.getSelectionModel().getSelectedItem() != null){

lblPersonOrt.setText(cbPersonPlz.getSelectionModel().getSelectedItem().getO
rt());
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
try {
        client = ClientRMI.getInstance();

    } catch (Exception e) {
        e.printStackTrace();
    }

        for (Plz plz :
FXCollections.observableArrayList(client.getAllPlz())) {
            cbPersonPlz.getItems().add(plz);
        }
```

```
        for (Subunternehmen subunternehmen :
FXCollections.observableArrayList(client.getAllSubunternehmen())) {
            cbPersonUnternehmen.getItems().add(subunternehmen);
        }
        for (Rolle rolle :
FXCollections.observableArrayList(client.getAllRolle())) {
            cbPersonRolle.getItems().add(rolle);
        }

cbPersonFunktion.setItems(FXCollections.observableArrayList("Bauherr","SuMi
tarbeiter","GuMitarbeiter"));
    }


}
```

AddProjektController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.Projektstatus;

public class AddProjektController implements Initializable {
        //RMI Client to interact
        ClientRMI client = null;
        RootController rootController = null;
        Projekt projekt = null;
        DateFormat formatDatum = null;
        DateTimeFormatter dateFormatter = null;
        List<Bauherr> b = null;
        Adresse a = null;
        Projektstatus ps = null;

        public void setRootController(RootController rootController) {
```

```
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }
        @FXML
        private TextField txtProjektBezeichnung;
        @FXML
        private ChoiceBox<Bauherr> cbProjektBauherr;
        @FXML
        private TextField txtProjektStrasse;
        @FXML
        private ComboBox<Plz> cbProjektPlz;
        @FXML
        private Label lblProjektOrt;
        @FXML
        private ChoiceBox<Objekttyp> cbProjektObjekttyp;
        @FXML
        private ChoiceBox<Arbeitstyp> cbProjektArbeitstyp;
        @FXML
        private DatePicker dateProjektStartdatum;
        @FXML
        private DatePicker dateProjektFaellig;

        @FXML
        private void addProjekt(){
            b = new ArrayList<Bauherr>();
            b.add(cbProjektBauherr.getSelectionModel().getSelectedItem());

            a = new Adresse(txtProjektStrasse.getText(),
cbProjektPlz.getSelectionModel().getSelectedItem());
            projekt = new Projekt(a,txtProjektBezeichnung.getText(),b, new
GregorianCalendar(dateProjektStartdatum.getValue().getYear(),
dateProjektStartdatum.getValue().getMonthValue() -1,
dateProjektStartdatum.getValue().getDayOfMonth()),null,cbProjektObjekttyp.g
etSelectionModel().getSelectedItem(),cbProjektArbeitstyp.getSelectionModel(
).getSelectedItem(), new
GregorianCalendar(dateProjektFaellig.getValue().getYear(),
dateProjektFaellig.getValue().getMonthValue() -1,
dateProjektFaellig.getValue().getDayOfMonth()),ps);
            //client.addAdresse(a);
            client.addProjekt(projekt);

            try {
                // Load Projekt overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/AusseresProjekt.fxml"));
                AnchorPane projekte = (AnchorPane) loader.load();

                ProjektController projektController =
loader.<ProjektController>getController();
                projektController.setRootController(rootController);

                rootController.rootLayout.setCenter(projekte);

            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        @FXML
        private void cancelProjekt(){
            try {
                // Load Projekt overview.
```

```
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/AusseresProjekt.fxml"));
                AnchorPane projekte = (AnchorPane) loader.load();

                ProjektController projektController =
loader.<ProjektController>getController();
                projektController.setRootController(rootController);

                rootController.rootLayout.setCenter(projekte);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @FXML
        private void plzChange(){
            if (cbProjektPlz.getSelectionModel().getSelectedItem() !=
null){

lblProjektOrt.setText(cbProjektPlz.getSelectionModel().getSelectedItem().ge
tOrt());
            }
        }



        @Override
        public void initialize(URL arg0, ResourceBundle arg1) {
            // TODO Auto-generated method stub
            client = ClientRMI.getInstance();
            formatDatum = new SimpleDateFormat("dd.MM.yyyy");
            dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
            for (Bauherr bauherr :
FXCollections.observableArrayList(client.getAllBauherr())) {
                cbProjektBauherr.getItems().add(bauherr);
            }
            for (Plz plz :
FXCollections.observableArrayList(client.getAllPlz())) {
                cbProjektPlz.getItems().add(plz);
            }
            for (Objekttyp objekttyp :
FXCollections.observableArrayList(client.getAllObjekttyp())) {
                cbProjektObjekttyp.getItems().add(objekttyp);
            }
            for (Arbeitstyp arbeitstyp :
FXCollections.observableArrayList(client.getAllArbeitstyp())) {
                cbProjektArbeitstyp.getItems().add(arbeitstyp);
            }
            for (Projektstatus projektstatus :
FXCollections.observableArrayList(client.getAllProjektstatus())) {
                if(projektstatus.getBezeichnung().equals("Offen")){
                    ps = projektstatus;
                }
            }

            // Client interaction
            try {
                client = ClientRMI.getInstance();
            } catch (Exception e) {
                e.printStackTrace();
```

```
            }
        }

}
```

AddUnternehmenController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The AddUnternehmenController handles all interaction with unternehmen *
 *
 * @author lkuendig
 * @version 1.0
 *
 */

public class AddUnternehmenController implements Initializable {
    //RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }
    Subunternehmen subunternehmen = null;
    Adresse adresse = null;

    @FXML
    public Label lblUnternehmenId;
    @FXML
    public TextField txtUnternehmenName;
    @FXML
    public TextField txtUnternehmenTelefon;
    @FXML
    public TextField txtUnternehmenStrasse;
    @FXML
    public ComboBox<Plz> cbUnternehmenPlz;
    @FXML
    public Label lblUnternehmenOrt;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
```

```
        try {
            client = new ClientRMI().getInstance();
            for (Plz plz :
FXCollections.observableArrayList(client.getAllPlz())) {
                cbUnternehmenPlz.getItems().add(plz);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }


    @FXML
    private void plzChange(){
        if (cbUnternehmenPlz.getSelectionModel().getSelectedItem() !=
null){

lblUnternehmenOrt.setText(cbUnternehmenPlz.getSelectionModel().getSelectedI
tem().getOrt());
        }
    }
    @FXML
    private void unternehmenSave() {

        adresse = new
Adresse(txtUnternehmenStrasse.getText(),cbUnternehmenPlz.getSelectionModel(
).getSelectedItem());
        subunternehmen = new Subunternehmen(adresse,
txtUnternehmenName.getText(), txtUnternehmenTelefon.getText());
        client.addSubunternehmen(subunternehmen);

        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
            subunternehmenController.setRootController(rootController);

            rootController.rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    @FXML
    public void unternehmenCancel(){
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
```

```
            subunternehmenController.setRootController(rootController);

            rootController.rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }


}
```

MangelController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.Writer;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;
import java.util.ResourceBundle;



import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.rmi.server.RMIServer;

/**
 * The MangelController handles all interaction with Mangel *
 *
 * @author sritz & mmont
 * @version 1.0
 *
 */

public class MangelController implements Initializable {
    String csvpath;
```

```
    // RMI Client to interact
    ClientRMI client = null;



    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }
    @FXML
    private Label lblMangelExport;
    // Define overviewtable with columns
    @FXML
    private TableView<Mangel> tblMangel;
    @FXML
    private TableColumn<Mangel, String> colMangelId;
    @FXML
    private TableColumn<Mangel, String> colMangelBezeichnung;
    @FXML
    private TableColumn<Mangel, String> colMangelProjekt;
    @FXML
    private TableColumn<Mangel, String> colMangelErfassungsdatum;
    @FXML
    private TableColumn<Mangel, String> colMangelFaelligkeitsdatum;
    @FXML
    private TableColumn<Mangel, String> colMangelAbschlusszeit;
    @FXML
    private TableColumn<Mangel, String> colMangelMangelstatus;

    // Datalist for Tableview
    ObservableList<Mangel> data;

    // SetCellValueFactory from overviewtable
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        csvpath = System.getProperty("user.home");
        lblMangelExport.setText("The File will be saved at: " + csvpath);


        DateFormat formatDatum = new SimpleDateFormat("dd.MM.yyyy");
        DateFormat formatZeit = new SimpleDateFormat("dd.MM.yyyy hh:mm");

        colMangelId
                .setCellValueFactory(new PropertyValueFactory<Mangel,
String>(
                        "id"));
        colMangelBezeichnung
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getBezeichnung());
                    }
                });
        colMangelProjekt
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        return new SimpleStringProperty(p.getValue()
```

```
                                .getFkProjekt().getBezeichnung());
                    }
                });

        colMangelErfassungsdatum
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        if (p.getValue().getErfassungsZeit() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new
SimpleStringProperty(formatZeit.format(p

.getValue().getErfassungsZeit().getTime()));
                        }
                    }
                });

        colMangelFaelligkeitsdatum
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        if (p.getValue().getFaelligkeitsDatum() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new SimpleStringProperty(formatDatum

.format(p.getValue().getFaelligkeitsDatum()
                                            .getTime()));
                        }
                    }
                });

        colMangelAbschlusszeit
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        if (p.getValue().getAbschlussZeit() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new
SimpleStringProperty(formatZeit.format(p

.getValue().getAbschlussZeit().getTime()));
                        }
                    }
                });

        colMangelMangelstatus
                .setCellValueFactory(new Callback<CellDataFeatures<Mangel,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Mangel, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkMangelstatus().getBezeichnung());
                    }
                });

        // Client interaction
```

```
        try {
            client = ClientRMI.getInstance();
            data =
FXCollections.observableArrayList(client.getAllMangel());
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Set data to tableview
        tblMangel.setItems(data);
    }

    /**
     * prints the TableView into a .csv file
     *
     * @throws IOException
     */
    public void exportTableView() throws IOException {

        DateFormat formatZeit = new SimpleDateFormat("dd.MM.yyyy_hh_mm");
        Writer writer = null;
        try {
            lblMangelExport.setText("The File was saved at: " + csvpath);
            client = ClientRMI.getInstance();
            data =
FXCollections.observableArrayList(client.getAllMangel());
            /*
             * Creates a CSV File in which the List will be saved C: is the
             * directory in which the File will be saved
             */
            File file = new File(csvpath + "\\" + (formatZeit.format(new
Date())) + "_Mangelliste.csv");
            writer = new BufferedWriter(new FileWriter(file));

            // The data that has to be put into the .csv File
            for (Mangel mangel : data) {
                String text = mangel.getId()
                        + ";"
                        + mangel.getFkProjekt().getBezeichnung()
                        + ";"
                        + mangel.getBezeichnung()
                        + ";"
                        + formatZeit.format(mangel.getErfassungsZeit()
                                .getTime())
                        + ";"
                        + formatZeit.format(mangel.getFaelligkeitsDatum()
                                .getTime()) + ";"
                        + mangel.getFkMangelstatus().getBezeichnung() + ";"
                        + "\n";

                writer.write(text);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {

            writer.flush();
            writer.close();
        }
    }

    @FXML
    public void showMangelDetail(MouseEvent t) throws IOException {
```

```
        if (t.getClickCount() == 2) {


            try {
                // Load MangelDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/mangel/InnererMangel.fxml"));
                AnchorPane innererMangel = (AnchorPane) loader.load();

                MangelDetailController detailMangelController = loader
                        .<MangelDetailController> getController();
                detailMangelController.setRootController(rootController);

                detailMangelController.init(tblMangel.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(innererMangel);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

MangelDetailController

```
package ch.hsluw.mangelmanager.client.intern.controller;

    import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;

    public class MangelDetailController implements Initializable {

        //RMI Client to interact
            ClientRMI client = null;
            RootController rootController = null;
            DateFormat formatDatum = null;
            DateTimeFormatter dateFormatter = null;

            public void setRootController(RootController rootController) {
```

```
                // TODO Auto-generated method stub
                this.rootController = rootController;
            }

            Mangel mangel = null;
            List<Mangelstatus> mangelstatusl = null;
            Mangelstatus mangelstatus = null;

            @FXML
            public Label lblMangelId;
            @FXML
            public TextArea txtMangelBeschreibung;
            @FXML
            public TextField txtMangelDatumanfang;
            @FXML
            public Label lblMangelFaellig;
            @FXML
            public DatePicker dateMangelDatumende;
            @FXML
            public Label lblMangelStatus;
            @FXML
            public Label lblMangelBezeichung;


            public void initialize(URL location, ResourceBundle resources)
{

            }

            public void init(Integer MangelId) {
                    try {
                    client = ClientRMI.getInstance();
                    mangel = client.getMangelById(MangelId);
                    mangelstatusl = client.getAllMangelStatus();
                    for (Mangelstatus mangelstatus : mangelstatusl) {
                        ;

if(mangelstatus.getBezeichnung().equals("Abgeschlossen")){
                                this.mangelstatus = mangelstatus;
                        }
                    }

                    formatDatum = new SimpleDateFormat("dd.MM.yyyy");
                    dateFormatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy");

                    lblMangelId.setText(mangel.getId().toString());
                    lblMangelBezeichung.setText(mangel.getBezeichnung());

lblMangelFaellig.setText((formatDatum.format(mangel.getFaelligkeitsDatum().
getTime())));

txtMangelBeschreibung.setText(mangel.getBeschreibung());

txtMangelDatumanfang.setText((formatDatum.format(mangel.getErfassungsZeit()
.getTime())));

                    if(mangel.getAbschlussZeit() == null){
                        dateMangelDatumende.setValue(null);
                    }else{

dateMangelDatumende.setValue(LocalDate.parse(formatDatum.format(mangel.getA
bschlussZeit().getTime()), dateFormatter));
```

```
                    }

lblMangelStatus.setText(mangel.getFkMangelstatus().getBezeichnung());


                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            @FXML
            private void mangelClose() {
                mangel.setFkMangelstatus(mangelstatus);
                if(dateMangelDatumende.getValue() != null){
                    mangel.setAbschlussZeit(new
GregorianCalendar(dateMangelDatumende.getValue().getYear(),
dateMangelDatumende.getValue().getMonthValue() -1,
dateMangelDatumende.getValue().getDayOfMonth()));
                }
                lblMangelStatus.setText("Abgeschlossen");
                client.updateMangel(mangel);

            }
            @FXML
            public void mangelCancel(){
                try {
                    // Load Unternehmen overview.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class

.getResource("view/mangel/AussererMangel.fxml"));
                    AnchorPane mangel = (AnchorPane) loader.load();

                    MangelController mangelController =
loader.<MangelController>getController();
                    mangelController.setRootController(rootController);

                    rootController.rootLayout.setCenter(mangel);

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

}
```

MeldungController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Meldung;

/**
 * The MeldungController handles all interaction with meldungen *
 *
 * @author lkuendig
 * @version 1.0
 *
 */

public class MeldungController implements Initializable {

    //RMI Client to interact
        ClientRMI client = null;

        RootController rootController = null;

        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        //Define overviewtable with columns
        @FXML
        private TableView<Meldung> tblMeldung;
        @FXML
        private TableColumn<Meldung, String> colMeldungId;
        @FXML
        private TableColumn<Meldung, String> colMeldungProjekt;
        @FXML
        private TableColumn<Meldung, String> colMeldungMangel;
        @FXML
        private TableColumn<Meldung, String> colMeldungTyp;
        @FXML
        private TableColumn<Meldung, String> colMeldungErfasst;
        @FXML
        private TableColumn<Meldung, String> colMeldungQuittiert;

        //Datalist for Tableview
        ObservableList<Meldung> data;

        //SetCellValueFactory from overviewtable
        @Override
        public void initialize(URL location, ResourceBundle resources) {

            DateFormat formatDatum = new SimpleDateFormat("dd.MM.yyyy");

            colMeldungId.setCellValueFactory(new
PropertyValueFactory<Meldung, String>("id"));
            colMeldungProjekt.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
```

```
                    return new
SimpleStringProperty(p.getValue().getFkMangel().getFkProjekt().getBezeichnu
ng());
                }
            });
            colMeldungMangel.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    return new
SimpleStringProperty(String.valueOf(p.getValue().getFkMangel().getId()));
                }
            });
            colMeldungTyp.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    return new
SimpleStringProperty(p.getValue().getFkMeldungstyp().getBezeichnung());
                }
            });
            colMeldungErfasst.setCellValueFactory(new
Callback<CellDataFeatures<Meldung, String>, ObservableValue<String>>() {
                public ObservableValue<String>
call(CellDataFeatures<Meldung, String> p) {
                    if (p.getValue().getZeitpunkt() == null){
                        return new SimpleStringProperty(" ");
                    }else{
                        return new
SimpleStringProperty(formatDatum.format(p.getValue().getZeitpunkt().getTime
()));
                    }
                }
            });
            colMeldungQuittiert.setCellValueFactory(new
PropertyValueFactory<Meldung, String>("quittiert"));

            //Client interaction
            try {
                client = new ClientRMI();
                data =
FXCollections.observableArrayList(client.getAllMeldung());
            } catch (Exception e) {
                e.printStackTrace();
            }

            //Set data to tableview
            tblMeldung.setItems(data);

        }

        @FXML
        public void showMeldungDetail(MouseEvent t) throws IOException{
            if(t.getClickCount() == 2){

                try {
                    // Load MeldungDetail View.
                    FXMLLoader loader = new FXMLLoader();
                    loader.setLocation(Main.class

.getResource("view/meldung/InnereMeldung.fxml"));
                    AnchorPane inneresMeldung = (AnchorPane) loader.load();
```

```
                    MeldungDetailController detailMeldungController =
loader.<MeldungDetailController>getController();

detailMeldungController.setRootController(rootController);


detailMeldungController.init(tblMeldung.getSelectionModel().getSelectedItem
().getId());
                    rootController.rootLayout.setCenter(inneresMeldung);


                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        }
}
```

MeldungDetailController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.AnchorPane;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Meldung;

public class MeldungDetailController implements Initializable {

    //RMI Client to interact
        ClientRMI client = null;
        RootController rootController = null;
        DateFormat formatZeit;


        public void setRootController(RootController rootController) {
            // TODO Auto-generated method stub
            this.rootController = rootController;
        }

        Meldung meldung = null;

        @FXML
        public Label lblMeldungId;
        @FXML
        public Label lblMeldungProjekt;
        @FXML
        public Label lblMeldungMangel;
        @FXML
        public Label lblMeldungDatum;
```

```
        @FXML
        public Label lblMeldungArt;
        @FXML
        public TextArea txtMeldungBeschreibung;

        @Override
        public void initialize(URL location, ResourceBundle resources) {
            formatZeit = new SimpleDateFormat("dd.MM.yyyy hh:mm");

        }


        public void init(int meldungId) {
                try {
                client = ClientRMI.getInstance();
                meldung = client.getMeldungById(meldungId);
                lblMeldungId.setText(meldung.getId().toString());

lblMeldungProjekt.setText(meldung.getFkMangel().getFkProjekt().getBezeichnu
ng());

lblMeldungMangel.setText(meldung.getFkMangel().getBezeichnung());

lblMeldungDatum.setText(formatZeit.format(meldung.getZeitpunkt().getTime())
);

lblMeldungArt.setText(meldung.getFkMeldungstyp().getBezeichnung());
                txtMeldungBeschreibung.appendText(meldung.getText());

            } catch (Exception e) {
                e.printStackTrace();
            }

        }

        @FXML
        private void meldungCancel() {
            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        @FXML
        public void meldungRead(){
            meldung.setQuittiert(true);
            client.updateMeldung(meldung);

            try {
                // Load Unternehmen overview.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
```

```
                        .getResource("view/meldung/AussereMeldung.fxml"));
                AnchorPane meldung = (AnchorPane) loader.load();

                MeldungController meldungController =
loader.<MeldungController>getController();
                meldungController.setRootController(rootController);

                rootController.rootLayout.setCenter(meldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

PersonController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;

/**
 * The PersonController handle all interactions with Persons
 *
 * @author Sandro
 * @version 1.0
 */
public class PersonController implements Initializable {
    RootController rootController = null;
    // RMI Client to interact
    ClientRMI client = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub

        this.rootController = rootController;
    }
```

```
    // Define overviewtable with columns
    @FXML
    private TableView<Person> tblPerson;
    @FXML
    private TableColumn<Person, String> colPersonId;
    @FXML
    private TableColumn<Person, String> colPersonName;
    @FXML
    private TableColumn<Person, String> colPersonVorname;
    @FXML
    private TableColumn<Person, String> colPersonTyp;
    @FXML
    private TableColumn<Person, String> colPersonUnternehmen;
    @FXML
    private TableColumn<Person, String> colPersonTelefon;
    @FXML
    private TableColumn<Person, String> colPersonBenutzername;
    @FXML
    private TableColumn<Person, String> colPersonEmail;
    @FXML
    private TableColumn<Person, String> colPersonRolle;

    // Datalist for Tableview
    ObservableList<Person> data;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        setCellValueFactoryTblPerson();

        // Client interaction
        try {
            client = ClientRMI.getInstance();
            data =
FXCollections.observableArrayList(client.getAllPerson());
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Set data to tableview
        tblPerson.setItems(data);
    }

    private void setCellValueFactoryTblPerson() {
        colPersonId
                .setCellValueFactory(new PropertyValueFactory<Person,
String>(
                        "id"));
        colPersonName
                .setCellValueFactory(new PropertyValueFactory<Person,
String>(
                        "nachname"));
        colPersonVorname
                .setCellValueFactory(new PropertyValueFactory<Person,
String>(
                        "vorname"));
        colPersonTyp
                .setCellValueFactory(new Callback<CellDataFeatures<Person,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Person, String> p) {
                        if (p.getValue() instanceof Bauherr) {
                            return new SimpleStringProperty("Bauherr");
                        } else if (p.getValue() instanceof GuMitarbeiter) {
```

```
                            return new SimpleStringProperty(
                                    "General-Mitarbeiter");
                        } else if (p.getValue() instanceof SuMitarbeiter) {
                            return new SimpleStringProperty(
                                    "Subunternehmen-Mitarbeiter");
                        } else {
                            return new SimpleStringProperty("unbekannt");
                        }

                    }
                });

        colPersonUnternehmen
                .setCellValueFactory(new Callback<CellDataFeatures<Person,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Person, String> p) {
                        if (p.getValue() instanceof Bauherr) {
                            return new SimpleStringProperty("Kein
Unternehmen");
                        } else if (p.getValue() instanceof GuMitarbeiter) {
                            return new SimpleStringProperty("W & W");
                        } else if (p.getValue() instanceof SuMitarbeiter) {
                            return new
SimpleStringProperty(((SuMitarbeiter) p
                                    .getValue()).getFkSubunternehmen()
                                    .getName());
                        } else {
                            return new SimpleStringProperty("unbekannt");
                        }

                    }
                });

        colPersonTelefon
                .setCellValueFactory(new PropertyValueFactory<Person,
String>(
                        "telefon"));
        colPersonBenutzername
                .setCellValueFactory(new Callback<CellDataFeatures<Person,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Person, String> p) {
                        if (p.getValue() instanceof Bauherr) {
                            return new SimpleStringProperty("-");
                        } else if (p.getValue() instanceof GuMitarbeiter) {
                            return new
SimpleStringProperty(((GuMitarbeiter) p

.getValue()).getFkLogin().getBenutzername());
                        } else if (p.getValue() instanceof SuMitarbeiter) {
                            return new
SimpleStringProperty(((SuMitarbeiter) p

.getValue()).getFkLogin().getBenutzername());
                        } else {
                            return new SimpleStringProperty("unbekannt");
                        }
                    }
                });
        colPersonEmail
                .setCellValueFactory(new Callback<CellDataFeatures<Person,
String>, ObservableValue<String>>() {
```

```
                    public ObservableValue<String> call(
                            CellDataFeatures<Person, String> p) {
                        if (p.getValue() instanceof Bauherr) {
                            return new SimpleStringProperty("-");
                        } else if (p.getValue() instanceof GuMitarbeiter) {
                            return new
SimpleStringProperty(((GuMitarbeiter) p
                                    .getValue()).getFkLogin().getEmail());
                        } else if (p.getValue() instanceof SuMitarbeiter) {
                            return new
SimpleStringProperty(((SuMitarbeiter) p
                                    .getValue()).getFkLogin().getEmail());
                        } else {
                            return new SimpleStringProperty("unbekannt");
                        }

                    }
                });
        colPersonRolle
                .setCellValueFactory(new Callback<CellDataFeatures<Person,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Person, String> p) {
                        if (p.getValue() instanceof Bauherr) {
                            return new SimpleStringProperty("-");
                        } else if (p.getValue() instanceof GuMitarbeiter) {
                            return new
SimpleStringProperty(((GuMitarbeiter) p
                                    .getValue()).getFkLogin().getFkrolle()
                                    .getName());
                        } else if (p.getValue() instanceof SuMitarbeiter) {
                            return new
SimpleStringProperty(((SuMitarbeiter) p
                                    .getValue()).getFkLogin().getFkrolle()
                                    .getName());
                        } else {
                            return new SimpleStringProperty("unbekannt");
                        }

                    }
                });
    }

    @FXML
    public void showPersonDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/person/InnerePerson.fxml"));
                AnchorPane inneresPerson = (AnchorPane) loader.load();

                PersonDetailController detailPersonController = loader
                        .<PersonDetailController> getController();
                detailPersonController.setRootController(rootController);

                detailPersonController.init(tblPerson.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresPerson);

            } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }
    }

    @FXML
    private void addPerson() {
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AddPerson.fxml"));
            AnchorPane addPerson = (AnchorPane) loader.load();

            AddPersonController addPersonController = loader
                    .<AddPersonController> getController();
            addPersonController.setRootController(rootController);

            rootController.rootLayout.setCenter(addPerson);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void deletePerson() {

    }
}
```

PersonDetailController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.TitledPane;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
```

```java
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;

/**
 * The ProjektDetailController handles all interaction with person *
 *
 * @author mmont
 * @version 1.0
 *
 */
public class PersonDetailController implements Initializable {
    // RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;
    DateFormat formatDatum = null;
    DateTimeFormatter dateFormatter = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Person person = null;

    // Left Panel
    @FXML
    private Label lblPersonId;
    @FXML
    private Label lblPersonName;
    @FXML
    private Label lblPersonVorname;
    @FXML
    private Label lblPersonUnternehmen;
    @FXML
    private TextField txtPersonStrasse;
    @FXML
    private ComboBox<Plz> cbPersonPlz;
    @FXML
    private Label lblPersonOrt;
    @FXML
    private TextField txtPersonTelefon;
    @FXML
    private Label lblPersonUnternehmenanz;

    // Right Panel
    @FXML
    private TitledPane tpPersonLogin;
    @FXML
    private Label lblPersonBenutzername;
    @FXML
    private Label lblPersonLoginRolle;
    @FXML
    private TextField txtPersonEmail;
    @FXML
    private TableView<Projekt> tblPersonProjekt;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektid;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektbezeichnung;
```

```
    @FXML
    private TableColumn<Projekt, String> colPersonProjektbauherr;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektadresse;
    @FXML
    private TableColumn<Projekt, String> colPersonProjektabgeschlossen;

    ObservableList<Projekt> projektData;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            client = ClientRMI.getInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }

        formatDatum = new SimpleDateFormat("dd.MM.yyyy");
        dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        for (Plz plz :
FXCollections.observableArrayList(client.getAllPlz())) {
            cbPersonPlz.getItems().add(plz);
        }

        setCellValueFactoryTblProjekt();
    }

    private void setCellValueFactoryTblProjekt() {
        colPersonProjektid
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "id"));
        colPersonProjektbezeichnung
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "bezeichnung"));
        colPersonProjektbauherr
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkBauherr().get(0).getNachname()
                                + " "
                                + p.getValue().getFkBauherr().get(0)
                                        .getVorname());
                    }
                });

        colPersonProjektadresse
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getStrasse()
                                + " "
                                +
p.getValue().getFkAdresse().getPlz().getPlz()
                                + " "
                                +
p.getValue().getFkAdresse().getPlz().getOrt());
                    }
```

```
                });
        colPersonProjektabgeschlossen
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkProjektstatus().getBezeichnung());
                    }
                });
    }

    public void init(int personId) {
        try {
            person = client.getPersonById(personId);
            projektData = FXCollections.observableArrayList(client
                    .getProjektbyPerson(person));
            lblPersonId.setText(person.getId().toString());
            lblPersonVorname.setText(person.getVorname());
            lblPersonName.setText(person.getNachname());

            if (person instanceof Bauherr) {
                cbPersonPlz.setDisable(false);
                lblPersonOrt.setDisable(false);
                txtPersonStrasse.setDisable(false);
                cbPersonPlz.getSelectionModel().select(
                        ((Bauherr)
person).getFkAdresse().getPlz().getPlz());
                cbPersonPlz.setPromptText(((Bauherr) person).getFkAdresse()
                        .getPlz().getPlz().toString());
                lblPersonOrt.setText(((Bauherr)
person).getFkAdresse().getPlz()
                        .getOrt());
                txtPersonStrasse.setText(((Bauherr) person).getFkAdresse()
                        .getStrasse());
            }
            txtPersonTelefon.setText(person.getTelefon());
            if (person instanceof GuMitarbeiter
                    || person instanceof SuMitarbeiter) {
                tpPersonLogin.setVisible(true);
                if (person instanceof SuMitarbeiter) {
                    lblPersonUnternehmenanz.setVisible(true);
                    lblPersonUnternehmen.setVisible(true);
                    lblPersonUnternehmenanz.setText(((SuMitarbeiter)
person)
                            .getFkSubunternehmen().getName());
                    lblPersonBenutzername.setText(((SuMitarbeiter) person)
                            .getFkLogin().getBenutzername());
                    txtPersonEmail.setText(((SuMitarbeiter) person)
                            .getFkLogin().getEmail());
                    lblPersonLoginRolle.setText(((SuMitarbeiter) person)
                            .getFkLogin().getFkrolle().getName());
                }
                if (person instanceof GuMitarbeiter) {
                    lblPersonBenutzername.setText(((GuMitarbeiter) person)
                            .getFkLogin().getBenutzername());
                    txtPersonEmail.setText(((GuMitarbeiter) person)
                            .getFkLogin().getEmail());
                    lblPersonLoginRolle.setText(((GuMitarbeiter) person)
                            .getFkLogin().getFkrolle().getName());
                }
            }
            tblPersonProjekt.setItems(projektData);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    @FXML
    public void showProjektDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {
            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/InneresProjekt.fxml"));
                AnchorPane inneresProjekt = (AnchorPane) loader.load();

                ProjektDetailController detailProjektController = loader
                        .<ProjektDetailController> getController();
                detailProjektController.setRootController(rootController);

                detailProjektController.init(tblPersonProjekt
                        .getSelectionModel().getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresProjekt);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

    @FXML
    private void plzChange() {
        if (cbPersonPlz.getSelectionModel().getSelectedItem() != null) {
            lblPersonOrt.setText(cbPersonPlz.getSelectionModel()
                    .getSelectedItem().getOrt());
        }
    }

    @FXML
    public void personCancel() {
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
            AnchorPane person = (AnchorPane) loader.load();

            PersonController personController = loader
                    .<PersonController> getController();
            personController.setRootController(rootController);

            rootController.rootLayout.setCenter(person);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void personSave() {
        if (person instanceof SuMitarbeiter) {
```

```
            person.setTelefon(txtPersonTelefon.getText());
            ((SuMitarbeiter) person).getFkLogin().setEmail(
                    txtPersonEmail.getText());
        } else if (person instanceof GuMitarbeiter) {
            person.setTelefon(txtPersonTelefon.getText());
            ((GuMitarbeiter) person).getFkLogin().setEmail(
                    txtPersonEmail.getText());
        } else if (person instanceof Bauherr) {
            ((Bauherr) person).getFkAdresse().setStrasse(
                    txtPersonStrasse.getText());
            ((Bauherr) person).getFkAdresse().setPlz(
                    cbPersonPlz.getSelectionModel().getSelectedItem());
            person.setTelefon(txtPersonTelefon.getText());
        }
        client.updatePerson(person);
    }
}
```

ProjektController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Projekt;

/**
 * The ProjektController handles all interaction with projects *
 *
 * @author sritz
 * @version 1.0
 *
 */

public class ProjektController implements Initializable {
    // RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
```

```
        this.rootController = rootController;
    }

    // Define overviewtable with columns
    @FXML
    private TableView<Projekt> tblProjekt;
    @FXML
    private TableColumn<Projekt, String> colProjektId;
    @FXML
    private TableColumn<Projekt, String> colProjektBezeichnung;
    @FXML
    private TableColumn<Projekt, String> colProjektBauherr;
    @FXML
    private TableColumn<Projekt, String> colProjektAdresse;
    @FXML
    private TableColumn<Projekt, String> colProjektOffeneMaengel;
    @FXML
    private TableColumn<Projekt, String> colProjektOffeneMeldungen;
    @FXML
    private TableColumn<Projekt, String> colProjektAbgeschlossen;

    // Datalist for Tableview
    ObservableList<Projekt> data;

    // Suche
    @FXML
    private ChoiceBox<String> cbProjektSearch;
    @FXML
    private TextField txtProjektSearch;

    @Override
    public void initialize(URL location, ResourceBundle resources) {

        setCellValueFactoryTblProjekt();

        // Client interaction
        try {
            client = ClientRMI.getInstance();
            data =
FXCollections.observableArrayList(client.getAllProjekt());
        } catch (Exception e) {
            e.printStackTrace();
        }

        cbProjektSearch.getItems().addAll(
                FXCollections.observableArrayList("Bezeichnung", "Bauherr",
                        "Plz", "Ort",
                        "Projektstatus"));
        cbProjektSearch.getSelectionModel().selectFirst();

        //If selected Item is changed clean txtProjektSearch
        cbProjektSearch.valueProperty().addListener(new
ChangeListener<String>() {
            @Override
            public void changed(ObservableValue<? extends String> ov,
String t, String t1) {
                txtProjektSearch.setText("");
            }
        });
        // Handle TextField text changes.
        txtProjektSearch.textProperty().addListener(new
ChangeListener<String>() {
            @Override
```

```
            public void changed(final ObservableValue<? extends String>
observable, final String oldValue, final String newValue) {
                if(newValue.length() <1){

updateTblProjekt(FXCollections.observableArrayList(client.getAllProjekt()))
;
                }
                else{
                switch
(cbProjektSearch.getSelectionModel().getSelectedItem()) {
                    case "Bezeichnung":

updateTblProjekt(FXCollections.observableArrayList(client.getProjektByBezei
chnung(txtProjektSearch.getText())));
                        break;
                    case "Bauherr":

updateTblProjekt(FXCollections.observableArrayList(client.getProjektByBauhe
rr(txtProjektSearch.getText())));
                        break;
                    case "Plz":

updateTblProjekt(FXCollections.observableArrayList(client.getProjektByPlz(t
xtProjektSearch.getText())));
                        break;
                    case "Ort":

updateTblProjekt(FXCollections.observableArrayList(client.getProjektByOrt(t
xtProjektSearch.getText())));
                        break;
                    case "Projektstatus":

updateTblProjekt(FXCollections.observableArrayList(client.getProjektByProje
ktstatus(txtProjektSearch.getText())));
                        break;
                    default:
                        updateTblProjekt(data);
                        break;
                    }
                }
            }

        });

        // Set data to tableview
        updateTblProjekt(data);
    }

    private void updateTblProjekt(ObservableList<Projekt> data) {
        // TODO Auto-generated method stub
            tblProjekt.setItems(data);
    }

    private void setCellValueFactoryTblProjekt() {
        // SetCellValueFactory from overviewtable
        colProjektId
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "id"));
        colProjektBezeichnung
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "bezeichnung"));
```

```
        colProjektBauherr
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkBauherr().get(0).getNachname()
                                + " "
                                + p.getValue().getFkBauherr().get(0)
                                        .getVorname());
                    }
                });

        colProjektAdresse
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getStrasse()
                                + " "
                                +
p.getValue().getFkAdresse().getPlz().getPlz()
                                + " "
                                +
p.getValue().getFkAdresse().getPlz().getOrt());
                    }
                });


        colProjektAbgeschlossen
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkProjektstatus().getBezeichnung());
                    }
                });
    }

    @FXML
    public void showProjektDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {
            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/InneresProjekt.fxml"));
                AnchorPane inneresProjekt = (AnchorPane) loader.load();

                ProjektDetailController detailProjektController = loader
                        .<ProjektDetailController> getController();
                detailProjektController.setRootController(rootController);

                detailProjektController.init(tblProjekt.getSelectionModel()
                        .getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresProjekt);

            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
        }

    }
    @FXML
    private void addProjekt(){
        try {
            // Load ProjektAdd View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AddProjekt.fxml"));
            AnchorPane addProjekt = (AnchorPane) loader.load();

            AddProjektController addProjektController = loader
                    .<AddProjektController> getController();
            addProjektController.setRootController(rootController);

            rootController.rootLayout.setCenter(addProjekt);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

ProjektDetailController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Mangel;
```

```java
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The ProjektDetailController handles all interaction with projects *
 *
 * @author sritz
 * @version 1.0
 *
 */

public class ProjektDetailController implements Initializable {
    // RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;
    DateFormat formatDatum = null;
    DateTimeFormatter dateFormatter = null;
    GregorianCalendar timestamp = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Projekt projekt = null;

    // Left Panel
    @FXML
    private Label lblPersonId;
    @FXML
    private Label lblProjektBauherr;
    @FXML
    private TextField txtProjektStrasse;
    @FXML
    private ComboBox<Plz> cbProjektPlz;
    @FXML
    private Label lblProjektOrt;
    @FXML
    private ComboBox<Objekttyp> cbProjektObjekttyp;
    @FXML
    private ComboBox<Arbeitstyp> cbProjektArbeitstyp;
    @FXML
    private Label lblProjektStartdatum;
    @FXML
    private DatePicker dateProjektEnddatum;
    @FXML
    private Label lblProjektFaellig;
    @FXML
    private ComboBox<Projektstatus> cbProjektStatus;

    // Right Panel Mangel
    @FXML
    private TableView<Mangel> tblProjektMangel;
    @FXML
    private TableColumn<Mangel, String> colProjektMangelId;
    @FXML
```

```
    private TableColumn<Mangel, String> colProjektMangelBezeichnung;
    ObservableList<Mangel> mangelData;

    // Right Panel Meldung
    @FXML
    private TableView<Meldung> tblProjektMeldung;
    @FXML
    private TableColumn<Meldung, String> colProjektMeldungId;
    @FXML
    private TableColumn<Meldung, String> colProjektMeldungBezeichnung;

    ObservableList<Meldung> meldungData;

    // Right Panel SubUnternehmen
    @FXML
    private TableView<Subunternehmen> tblProjektUnternehmen;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenId;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenName;
    @FXML
    private TableColumn<Subunternehmen, String>
colProjektUnternehmenStrasse;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenPlz;
    @FXML
    private TableColumn<Subunternehmen, String> colProjektUnternehmenOrt;
    @FXML
    private TableColumn<Subunternehmen, String>
colProjektUnternehmenTelefon;

    ObservableList<Subunternehmen> subunternehmenData;

    @FXML
    private ComboBox<Subunternehmen> cbSubunternehmen;
    @FXML
    private ComboBox<SuMitarbeiter> cbAnsprechperson;

    // Right Panel Bauleiter
    @FXML
    private TableView<ProjektGuMitarbeiter> tblProjektBauleiter;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterId;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterName;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterVorname;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterStartdatum;
    @FXML
    private TableColumn<ProjektGuMitarbeiter, String>
colProjektBauleiterEnddatum;

    @FXML
    private ComboBox<GuMitarbeiter> cbProjektBauleiter;

    ObservableList<ProjektGuMitarbeiter> bauleiterData;

    @Override
```

```
    public void initialize(URL location, ResourceBundle resources) {
        client = ClientRMI.getInstance();
        formatDatum = new SimpleDateFormat("dd.MM.yyyy");
        dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
        timestamp = (GregorianCalendar) Calendar.getInstance();

        for (Plz plz :
FXCollections.observableArrayList(client.getAllPlz())) {
            cbProjektPlz.getItems().add(plz);
        }
        for (Objekttyp objekttyp : FXCollections.observableArrayList(client
                .getAllObjekttyp())) {
            cbProjektObjekttyp.getItems().add(objekttyp);
        }
        for (Arbeitstyp arbeitstyp :
FXCollections.observableArrayList(client
                .getAllArbeitstyp())) {
            cbProjektArbeitstyp.getItems().add(arbeitstyp);
        }
        for (Subunternehmen subunternehmen : FXCollections
                .observableArrayList(client.getAllSubunternehmen())) {
            cbSubunternehmen.getItems().add(subunternehmen);
        }
        for (GuMitarbeiter guMitarbeiter : FXCollections
                .observableArrayList(client.getAllGuMitarbeiter())) {
            cbProjektBauleiter.getItems().add(guMitarbeiter);
        }
        for (Projektstatus projektStatus : FXCollections
                .observableArrayList(client.getAllProjektstatus())) {
            cbProjektStatus.getItems().add(projektStatus);
        }

        setCellValueFactoryTblMangel();
        setCellValueFactoryTblMeldung();
        setCellValueFactoryTblUnternehmen();
        setCellValueFactoryTblBauleiter();

    }

    private void setCellValueFactoryTblBauleiter() {
        // TODO Auto-generated method stub
        colProjektBauleiterId
                .setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(

TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkProjekt().getId().toString());
                    }
                });
        colProjektBauleiterName
                .setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(

TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkMitarbeiter().getNachname());
                    }
                });
```

```
        colProjektBauleiterVorname
                .setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(

TableColumn.CellDataFeatures<ProjektGuMitarbeiter, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkMitarbeiter().getVorname());
                    }
                });

        colProjektBauleiterStartdatum
                .setCellValueFactory(new
Callback<CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<ProjektGuMitarbeiter, String>
p) {
                        if (p.getValue().getStartDatum() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new SimpleStringProperty(formatDatum
                                    .format(p.getValue().getStartDatum()
                                            .getTime()));
                        }
                    }
                });
        colProjektBauleiterEnddatum
                .setCellValueFactory(new
Callback<CellDataFeatures<ProjektGuMitarbeiter, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<ProjektGuMitarbeiter, String>
p) {
                        if (p.getValue().getEndDatum() == null) {
                            return new SimpleStringProperty(" ");
                        } else {
                            return new SimpleStringProperty(formatDatum
                                    .format(p.getValue().getEndDatum()
                                            .getTime()));
                        }
                    }
                });
    }

    private void setCellValueFactoryTblMeldung() {
        colProjektMeldungId
                .setCellValueFactory(new PropertyValueFactory<Meldung,
String>(
                        "id"));
        colProjektMeldungBezeichnung
                .setCellValueFactory(new PropertyValueFactory<Meldung,
String>(
                        "text"));
    }

    private void setCellValueFactoryTblMangel() {
        colProjektMangelId
                .setCellValueFactory(new PropertyValueFactory<Mangel,
String>(
                        "id"));
        colProjektMangelBezeichnung
```

```
                .setCellValueFactory(new PropertyValueFactory<Mangel,
String>(
                        "bezeichnung"));
    }

    private void setCellValueFactoryTblUnternehmen() {
        colProjektUnternehmenId
                .setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>(
                        "id"));
        colProjektUnternehmenName
                .setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>(
                        "name"));
        colProjektUnternehmenStrasse
                .setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            TableColumn.CellDataFeatures<Subunternehmen,
String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getStrasse());
                    }
                });
        colProjektUnternehmenPlz
                .setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            TableColumn.CellDataFeatures<Subunternehmen,
String> p) {
                        return new SimpleStringProperty(p.getValue()

.getFkAdresse().getPlz().getPlz().toString());
                    }
                });
        colProjektUnternehmenOrt
                .setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            TableColumn.CellDataFeatures<Subunternehmen,
String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getPlz().getOrt());
                    }
                });
        colProjektUnternehmenTelefon
                .setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>(
                        "telefon"));
    }

    public void init(int projektId) {
        try {
            projekt = client.getProjektById(projektId);
            lblPersonId.setText(projekt.getId().toString());
            lblProjektBauherr.setText(projekt.getFkBauherr().get(0)
                    .getNachname()
                    + " " + projekt.getFkBauherr().get(0).getVorname());
            txtProjektStrasse.setText(projekt.getFkAdresse().getStrasse());
            cbProjektPlz.getSelectionModel().select(
```

```
                    projekt.getFkAdresse().getPlz());

cbProjektPlz.setPromptText(projekt.getFkAdresse().getPlz().getPlz()
                    .toString());
            lblProjektOrt.setText(cbProjektPlz.getSelectionModel()
                    .getSelectedItem().getOrt());
            cbProjektArbeitstyp.setValue(projekt.getFkArbeitstyp());
            cbProjektArbeitstyp.setPromptText(projekt.getFkArbeitstyp()
                    .getBezeichnung());
            cbProjektObjekttyp.getSelectionModel().select(
                    projekt.getFkObjekttyp());
            cbProjektObjekttyp.setPromptText(projekt.getFkObjekttyp()
                    .getBezeichnung());

            lblProjektStartdatum.setText(formatDatum.format(projekt
                    .getStartDatum().getTime()));
            if (projekt.getEndDatum() != null) {
                dateProjektEnddatum.setValue(LocalDate.parse(

formatDatum.format(projekt.getEndDatum().getTime()),
                        dateFormatter));

            }
            lblProjektFaellig.setText(formatDatum.format(projekt
                    .getFaelligkeitsDatum().getTime()));
            cbProjektStatus.getSelectionModel().select(
                    projekt.getFkProjektstatus());

            mangelData = FXCollections.observableArrayList(client
                    .getAllMangelProjekt(projekt.getId()));
            subunternehmenData = FXCollections.observableArrayList(client
                    .getUnternehmenByProjekt(projekt.getId()));
            bauleiterData = FXCollections.observableArrayList(client
                    .getBauleiterByProjekt(projekt));
            tblProjektMangel.setItems(mangelData);
            tblProjektUnternehmen.setItems(subunternehmenData);
            tblProjektBauleiter.setItems(bauleiterData);

        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    @FXML
    public void showMeldungByMangelOderMangel(MouseEvent t) throws
IOException {
        if (t.getClickCount() == 1) {
            meldungData = FXCollections.observableArrayList(client

.getAllMeldungByMangel(tblProjektMangel.getSelectionModel()
                            .getSelectedItem()));
            tblProjektMeldung.setItems(meldungData);
        } else if (t.getClickCount() == 2) {

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/mangel/InnererMangel.fxml"));
                AnchorPane innererMangel = (AnchorPane) loader.load();

                MangelDetailController mangelDetailController = loader
```

```
                    .<MangelDetailController> getController();
            mangelDetailController.setRootController(rootController);

            mangelDetailController.init(tblProjektMangel
                    .getSelectionModel().getSelectedItem().getId());
            rootController.rootLayout.setCenter(innererMangel);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}

@FXML
public void projektCancel() {
    try {
        // Load ProjektDetail View.
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(Main.class
                .getResource("view/projekt/AusseresProjekt.fxml"));
        AnchorPane ausseresProjekt = (AnchorPane) loader.load();

        ProjektController projektController = loader
                .<ProjektController> getController();
        projektController.setRootController(rootController);

        rootController.rootLayout.setCenter(ausseresProjekt);

    } catch (IOException e) {
        e.printStackTrace();
    }
}

@FXML
public void fillCbAnsprechperson() {
    for (SuMitarbeiter sumitarbeiter : client
            .getAllSubunternehmenMitarbeiter(cbSubunternehmen
                    .getSelectionModel().getSelectedItem())) {
        cbAnsprechperson.getItems().add(sumitarbeiter);
    }
}

@FXML
public void projektSave() {

    projekt.getFkAdresse().setStrasse(txtProjektStrasse.getText());
    projekt.getFkAdresse()
            .getPlz()
            .setPlz((Integer) cbProjektPlz.getSelectionModel()
                    .getSelectedItem().getPlz());
    projekt.getFkAdresse().getPlz().setOrt(lblProjektOrt.getText());

    projekt.setFkObjekttyp(cbProjektObjekttyp.getSelectionModel()
            .getSelectedItem());
    projekt.setFkArbeitstyp(cbProjektArbeitstyp.getSelectionModel()
            .getSelectedItem());
    if (dateProjektEnddatum.getValue() != null) {
        projekt.setEndDatum(new GregorianCalendar(dateProjektEnddatum
                .getValue().getYear(), dateProjektEnddatum.getValue()
                .getMonthValue() - 1, dateProjektEnddatum.getValue()
                .getDayOfMonth()));
    }
```

```
        projekt.setFkProjektstatus(cbProjektStatus.getSelectionModel()
                .getSelectedItem());
        client.updateProjekt(projekt);
        try {
            // Load Projekt overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AusseresProjekt.fxml"));
            AnchorPane projekte = (AnchorPane) loader.load();

            ProjektController projektController = loader
                    .<ProjektController> getController();
            projektController.setRootController(rootController);

            rootController.rootLayout.setCenter(projekte);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void projektAddMangel() {
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/mangel/AddMangel.fxml"));
            AnchorPane addMangel = (AnchorPane) loader.load();

            AddMangelController addMangelController = loader
                    .<AddMangelController> getController();
            addMangelController.setRootController(rootController);

            addMangelController.init(projekt);
            rootController.rootLayout.setCenter(addMangel);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void projektAddMeldung() {
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/meldung/AddMeldung.fxml"));
            AnchorPane addMeldung = (AnchorPane) loader.load();

            AddMeldungController addMeldungController = loader
                    .<AddMeldungController> getController();
            addMeldungController.setRootController(rootController);

            addMeldungController.init(tblProjektMangel.getSelectionModel()
                    .getSelectedItem());
            rootController.rootLayout.setCenter(addMeldung);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    @FXML
    private void showMeldungDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/meldung/InnereMeldung.fxml"));
                AnchorPane innereMeldung = (AnchorPane) loader.load();

                MeldungDetailController meldungDetailController = loader
                        .<MeldungDetailController> getController();
                meldungDetailController.setRootController(rootController);

                meldungDetailController.init(tblProjektMeldung
                        .getSelectionModel().getSelectedItem().getId());
                rootController.rootLayout.setCenter(innereMeldung);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    private void showSubunternehmenDetail(MouseEvent t) throws IOException
{
        if (t.getClickCount() == 2) {


            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class

.getResource("view/unternehmen/InneresUnternehmen.fxml"));
                AnchorPane inneresUnternehmen = (AnchorPane) loader.load();

                SubUnternehmenDetailController
subunternehmenDetailController = loader
                        .<SubUnternehmenDetailController> getController();
                subunternehmenDetailController
                        .setRootController(rootController);

                subunternehmenDetailController.init(tblProjektUnternehmen
                        .getSelectionModel().getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresUnternehmen);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    private void showBauleiterDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {


            try {
                // Load ProjektDetail View.
```

```
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/person/InnerePerson.fxml"));
                AnchorPane innerePerson = (AnchorPane) loader.load();

                PersonDetailController personDetailController = loader
                        .<PersonDetailController> getController();
                personDetailController.setRootController(rootController);

                personDetailController.init(tblProjektBauleiter
                        .getSelectionModel().getSelectedItem().getId());
                rootController.rootLayout.setCenter(innerePerson);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    public void projektAddUnternehmen() {
        client.addSuMitarbeiterByProjekt(new ProjektSuMitarbeiter(projekt,
                cbAnsprechperson.getSelectionModel().getSelectedItem(),
                timestamp, null));
        subunternehmenData = FXCollections.observableArrayList(client
                .getUnternehmenByProjekt(projekt.getId()));
        tblProjektUnternehmen.setItems(subunternehmenData);
    }

    @FXML
    public void projektAddBauleiter() {
        // Letzter Bauleiter Enddatum sezten
        ProjektGuMitarbeiter lastBauleiter = tblProjektBauleiter.getItems()
                .get(tblProjektBauleiter.getItems().size() - 1);
        lastBauleiter.setEndDatum(timestamp);
        client.updateProjektGuMitarbeiter(lastBauleiter);
        bauleiterData = FXCollections.observableArrayList(client
                .getBauleiterByProjekt(projekt));
        tblProjektBauleiter.setItems(bauleiterData);
        client.addGuMitarbeiterByProjekt(new ProjektGuMitarbeiter(projekt,
                cbProjektBauleiter.getSelectionModel().getSelectedItem(),
                timestamp, null));
        bauleiterData = FXCollections.observableArrayList(client
                .getBauleiterByProjekt(projekt));
        tblProjektBauleiter.setItems(bauleiterData);
    }

    @FXML
    private void plzChange() {
        if (cbProjektPlz.getSelectionModel().getSelectedItem() != null) {
            lblProjektOrt.setText(cbProjektPlz.getSelectionModel()
                    .getSelectedItem().getOrt());
        }
    }

}
```

RootController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
```

```java
import java.net.URL;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.Menu;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Login;

/**
 * The RootController is used to load different content
 * in center of the BorderPane
 *
 *
 * @author sritz, mmonti
 * @version 1.0
 *
 */
public class RootController implements Initializable {


    @FXML
    public BorderPane rootLayout;

    @FXML
    private Menu menuBenutzer;

    @FXML
    private void logout() {
        // Load Login and close current Stage
        Main.initRootLayout();
        Stage stageToClose = (Stage) rootLayout.getScene().getWindow();
        stageToClose.close();
        menuBenutzer.setText("");

    }
    @FXML
    private void showPersonen() {
        try {
            // Load Person overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/person/AusserePerson.fxml"));
            AnchorPane personen = (AnchorPane) loader.load();

            PersonController personController =
loader.<PersonController>getController();
            personController.setRootController(this);

            rootLayout.setCenter(personen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showUnternehmen() {
```

```
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController =
loader.<SubUnternehmenController>getController();
            subunternehmenController.setRootController(this);

            rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showProjekte() {
        try {
            // Load Projekt overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/projekt/AusseresProjekt.fxml"));
            AnchorPane projekte = (AnchorPane) loader.load();

            ProjektController projektController =
loader.<ProjektController>getController();
            projektController.setRootController(this);

            rootLayout.setCenter(projekte);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showMaengel() {
        try {
            // Load Maengel overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/mangel/AussererMangel.fxml"));
            AnchorPane maengel = (AnchorPane) loader.load();

            MangelController mangelController =
loader.<MangelController>getController();
            mangelController.setRootController(this);

            rootLayout.setCenter(maengel);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void showMeldungen() {
        try {
            // Load Meldung overview.
```

```
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/meldung/AussereMeldung.fxml"));
            AnchorPane meldungen = (AnchorPane) loader.load();

            MeldungController meldungController =
loader.<MeldungController>getController();
            meldungController.setRootController(this);

            rootLayout.setCenter(meldungen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        // TODO Auto-generated method stub
        showPersonen();
    }
    public void init() {
        // TODO Auto-generated method stub
        ClientRMI client = ClientRMI.getInstance();
        Login login = client.getLoginById(Main.loginId);
        menuBenutzer.setText("Benutzer: " +login.getBenutzername());

    }

}
```

SubUnternehmenController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * The SubunternehmenController handles all interaction with subunternehmen
*
 *
 * @author lkuendig
 * @version 1.0
```

```
 *
 */

public class SubUnternehmenController implements Initializable {
    //RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    //Define overviewtable with columns
    @FXML
    private TableView<Subunternehmen> tblSubunternehmen;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenId;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenName;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenAdresse;
    @FXML
    private TableColumn<Subunternehmen, String> colSubunternehmenTelefon;
    @FXML
    private TableColumn<Subunternehmen, String>
colSubunternehmenOffeneProjekte;

    //Datalist for Tableview
    ObservableList<Subunternehmen> data;


    //SetCellValueFactory from overviewtable
    @Override
    public void initialize(URL location, ResourceBundle resources) {

        colSubunternehmenId.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("id"));
        colSubunternehmenName.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("name"));

        colSubunternehmenAdresse.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
                return new
SimpleStringProperty(p.getValue().getFkAdresse().getStrasse() + " "
+p.getValue().getFkAdresse().getPlz().getPlz() + " " +
p.getValue().getFkAdresse().getPlz().getOrt());
            }
        });
        colSubunternehmenTelefon.setCellValueFactory(new
PropertyValueFactory<Subunternehmen, String>("telefon"));

        colSubunternehmenOffeneProjekte.setCellValueFactory(new
Callback<TableColumn.CellDataFeatures<Subunternehmen, String>,
ObservableValue<String>>() {
            public ObservableValue<String>
call(TableColumn.CellDataFeatures<Subunternehmen, String> p) {
                return new
SimpleStringProperty(client.getProjektproSubunternehmen(p.getValue().getId(
)));
```

```
            }
        });


        //Client interaction
        try {
            client = new ClientRMI();
            data =
FXCollections.observableArrayList(client.getAllSubunternehmen());
        } catch (Exception e) {
            e.printStackTrace();
        }


        //Set data to tableview
        tblSubunternehmen.setItems(data);
    }

    @FXML
    public void showSubunternehmenDetail(MouseEvent t) throws IOException{
        if(t.getClickCount() == 2){

            try {
                // Load SubunternehmenDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class

.getResource("view/unternehmen/InneresUnternehmen.fxml"));
                AnchorPane inneresUnternehmen = (AnchorPane) loader.load();

                SubUnternehmenDetailController
detailSubunternehmenController =
loader.<SubUnternehmenDetailController>getController();

detailSubunternehmenController.setRootController(rootController);


detailSubunternehmenController.init(tblSubunternehmen.getSelectionModel().g
etSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresUnternehmen);


            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    @FXML
    private void addUnternehmen(){
        try {
            // Load ProjektDetail View.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class
                    .getResource("view/unternehmen/AddUnternehmen.fxml"));
            AnchorPane addUnternehmen = (AnchorPane) loader.load();

            AddUnternehmenController addUnternehmenController =
loader.<AddUnternehmenController>getController();
            addUnternehmenController.setRootController(rootController);
            rootController.rootLayout.setCenter(addUnternehmen);


        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }

}
```

SubUnternehmenDetailController

```
package ch.hsluw.mangelmanager.client.intern.controller;

import java.io.IOException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ResourceBundle;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.util.Callback;
import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.client.intern.Main;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

public class SubUnternehmenDetailController implements Initializable {

    // RMI Client to interact
    ClientRMI client = null;
    RootController rootController = null;
    DateFormat formatDatum = null;
    DateTimeFormatter dateFormatter = null;

    public void setRootController(RootController rootController) {
        // TODO Auto-generated method stub
        this.rootController = rootController;
    }

    Subunternehmen subunternehmen = null;

    @FXML
    public Label lblUnternehmenId;
    @FXML
    public TextField txtUnternehmenName;
```

```
    @FXML
    public TextField txtUnternehmenTelefon;
    @FXML
    public TextField txtUnternehmenStrasse;
    @FXML
    public ComboBox<Plz> cbUnternehmenPlz;
    @FXML
    public Label lblUnternehmenOrt;

    // Projekte pro Subunternehmen
    @FXML
    private TableView<Projekt> tblUnternehmenProjekt;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektId;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektBezeichnung;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektBauherr;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektStrasse;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektPlz;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektOrt;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektStartdatum;
    @FXML
    private TableColumn<Projekt, String> colUnternehmenProjektStatus;

    // Projekte pro Subunternehmen
    @FXML
    private TableView<SuMitarbeiter> tblUnternehmenMitarbeiter;
    @FXML
    private TableColumn<SuMitarbeiter, String> colUnternehmenMitarbeiterId;
    @FXML
    private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterName;
    @FXML
    private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterVorname;
    @FXML
    private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterStartdatum;
    @FXML
    private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterEnddatum;
    @FXML
    private TableColumn<SuMitarbeiter, String>
colUnternehmenMitarbeiterTelefon;

    // Datalist for Tableview
    ObservableList<Projekt> data;
    ObservableList<SuMitarbeiter> data2;

    @Override
    public void initialize(URL location, ResourceBundle resources) {

        try {
            client = ClientRMI.getInstance();
            for (Plz plz : FXCollections
                    .observableArrayList(client.getAllPlz())) {
                cbUnternehmenPlz.getItems().add(plz);
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void init(int subunternehmenId) {
        setCellValueFactoryTblUnternehmenProjekt();
        setCellValueFactoryTblUnternehmenMiterbeiter();
        try {

            formatDatum = new SimpleDateFormat("dd.MM.yyyy");
            dateFormatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
            subunternehmen =
client.getSubunternehmenById(subunternehmenId);
            lblUnternehmenId.setText(subunternehmen.getId().toString());
            txtUnternehmenName.setText(subunternehmen.getName());
            txtUnternehmenTelefon.setText(subunternehmen.getTelefon());
            txtUnternehmenStrasse.setText(subunternehmen.getFkAdresse()
                    .getStrasse());

cbUnternehmenPlz.setValue(subunternehmen.getFkAdresse().getPlz());

lblUnternehmenOrt.setText(subunternehmen.getFkAdresse().getPlz()
                    .getOrt());

            data = FXCollections.observableArrayList(client
                    .getAllSubunternehmenProjekt(subunternehmen.getId()));
            data2 = FXCollections.observableArrayList(client
                    .getAllSubunternehmenMitarbeiter(subunternehmen));
            // Set data to tableview
            tblUnternehmenProjekt.setItems(data);
            tblUnternehmenMitarbeiter.setItems(data2);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    private void setCellValueFactoryTblUnternehmenProjekt() {
        // TODO Auto-generated method stub
        colUnternehmenProjektId
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "id"));
        colUnternehmenProjektBezeichnung
                .setCellValueFactory(new PropertyValueFactory<Projekt,
String>(
                        "bezeichnung"));

        colUnternehmenProjektBauherr
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkBauherr().get(0).getNachname()
                                + " "
                                + p.getValue().getFkBauherr().get(0)
                                        .getVorname());
                    }
                });
```

```
        colUnternehmenProjektStrasse
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getStrasse());
                    }
                });
        colUnternehmenProjektPlz
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()

.getFkAdresse().getPlz().getPlz().toString());
                    }
                });
        colUnternehmenProjektOrt
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkAdresse().getPlz().getOrt());
                    }
                });

        colUnternehmenProjektStartdatum
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new
SimpleStringProperty(formatDatum.format(p
                                .getValue().getStartDatum().getTime()));
                    }
                });

        colUnternehmenProjektStatus
                .setCellValueFactory(new Callback<CellDataFeatures<Projekt,
String>, ObservableValue<String>>() {
                    public ObservableValue<String> call(
                            CellDataFeatures<Projekt, String> p) {
                        return new SimpleStringProperty(p.getValue()
                                .getFkProjektstatus().getBezeichnung());
                    }
                });
    }

    public void setCellValueFactoryTblUnternehmenMiterbeiter() {
        colUnternehmenMitarbeiterName
                .setCellValueFactory(new
PropertyValueFactory<SuMitarbeiter, String>(
                        "nachname"));
        colUnternehmenMitarbeiterVorname
                .setCellValueFactory(new
PropertyValueFactory<SuMitarbeiter, String>(
                        "vorname"));
        colUnternehmenMitarbeiterTelefon
                .setCellValueFactory(new
PropertyValueFactory<SuMitarbeiter, String>(
```

```
                        "telefon"));
    }

    @FXML
    private void unternehmenSave() {

        subunternehmen.setName(txtUnternehmenName.getText());
        subunternehmen.setTelefon(txtUnternehmenTelefon.getText());
        subunternehmen.getFkAdresse().setPlz(
                cbUnternehmenPlz.getSelectionModel().getSelectedItem());
        subunternehmen.getFkAdresse().getPlz()
                .setOrt(lblUnternehmenOrt.getText());
        subunternehmen.getFkAdresse().setStrasse(
                txtUnternehmenStrasse.getText());
        client.updateSubunternehmen(subunternehmen);

        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController = loader
                    .<SubUnternehmenController> getController();
            subunternehmenController.setRootController(rootController);

            rootController.rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    @FXML
    private void unternehmenCancel() {
        try {
            // Load Unternehmen overview.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class

.getResource("view/unternehmen/AusseresUnternehmen.fxml"));
            AnchorPane unternehmen = (AnchorPane) loader.load();

            SubUnternehmenController subunternehmenController = loader
                    .<SubUnternehmenController> getController();
            subunternehmenController.setRootController(rootController);

            rootController.rootLayout.setCenter(unternehmen);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void plzChange() {
        if (cbUnternehmenPlz.getSelectionModel().getSelectedItem() != null)
{
            lblUnternehmenOrt.setText(cbUnternehmenPlz.getSelectionModel()
                    .getSelectedItem().getOrt());
```

```
        } else {
        }
    }

    @FXML
    private void showProjektDetail(MouseEvent t) throws IOException {
        if (t.getClickCount() == 2) {

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/projekt/InneresProjekt.fxml"));
                AnchorPane inneresProjekt = (AnchorPane) loader.load();

                ProjektDetailController detailProjektController = loader
                        .<ProjektDetailController> getController();
                detailProjektController.setRootController(rootController);

                detailProjektController.init(tblUnternehmenProjekt
                        .getSelectionModel().getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresProjekt);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    @FXML
    public void showMitarbeiterDetail(MouseEvent t) {
        if (t.getClickCount() == 2) {

            try {
                // Load ProjektDetail View.
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(Main.class
                        .getResource("view/person/InnerePerson.fxml"));
                AnchorPane inneresPerson = (AnchorPane) loader.load();

                PersonDetailController detailPersonController = loader
                        .<PersonDetailController> getController();
                detailPersonController.setRootController(rootController);

                detailPersonController.init(tblUnternehmenMitarbeiter
                        .getSelectionModel().getSelectedItem().getId());
                rootController.rootLayout.setCenter(inneresPerson);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

ClientRMITest

```
package ch.hsluw.mangelmanager.client.test;

import static org.junit.Assert.assertTrue;
import static org.junit.Assert.*;
```

```java
import java.util.GregorianCalendar;
import java.util.List;

import org.junit.Test;

import ch.hsluw.mangelmanager.client.intern.ClientRMI;
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
import ch.hsluw.mangelmanager.rmi.adresse.AdresseRO;
import ch.hsluw.mangelmanager.rmi.arbeitstyp.ArbeitstypRO;
import ch.hsluw.mangelmanager.rmi.bauherr.BauherrRO;
import ch.hsluw.mangelmanager.rmi.gumitarbeiter.GuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.login.LoginRO;
import ch.hsluw.mangelmanager.rmi.mangel.MangelRO;
import ch.hsluw.mangelmanager.rmi.mangelstatus.MangelstatusRO;
import ch.hsluw.mangelmanager.rmi.meldung.MeldungRO;
import ch.hsluw.mangelmanager.rmi.meldungstyp.MeldungstypRO;
import ch.hsluw.mangelmanager.rmi.objekttyp.ObjekttypRO;
import ch.hsluw.mangelmanager.rmi.person.PersonRO;
import ch.hsluw.mangelmanager.rmi.plz.PlzRO;
import ch.hsluw.mangelmanager.rmi.projekt.ProjektRO;
import
ch.hsluw.mangelmanager.rmi.projektgumitarbeiter.ProjektGuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.projektstatus.ProjektstatusRO;
import
ch.hsluw.mangelmanager.rmi.projektsumitarbeiter.ProjektSuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.rolle.RolleRO;
import ch.hsluw.mangelmanager.rmi.subunternehmen.SubunternehmenRO;
import ch.hsluw.mangelmanager.rmi.sumitarbeiter.SuMitarbeiterRO;

/**
 * Diese Klasse testet die Methoden von ClientRMI
 *
 * @version 1.0
 * @author mmont
 *
 */

public class ClientRMITest {

    // all the needed objects

    List<Person> person;
    List<Projekt> projekte;
    List<Projekt> suprojekte;
```

```
List<Projektstatus> projektstatus;
List<Subunternehmen> subunternehmen;

List<Mangel> maengel;
List<Meldung> meldung;
List<Plz> plz;
List<Objekttyp> objekttyp;
List<Arbeitstyp> arbeitstyp;
List<SuMitarbeiter> sumitarbeiter;
List<ProjektGuMitarbeiter> bauleiter;
List<Mangel> mangelOfProjekt;
List<Meldung> meldungByMangel;
List<Bauherr> bauherren;
List<GuMitarbeiter> guMitarbeiter;

List<Mangelstatus> mangelstatus;
List<Meldungstyp> meldungstyp;
String anzProjekte;
Projekt projekt;
Subunternehmen subunternehmennr;
Meldung meldungnr;
Plz plznr;
Mangel mangelnr;
Adresse addAdresse;
Login login;
Login loginnr;
Person personnr;
List<Rolle> rollen;

PersonRO personRO;
ProjektRO projektRO;
SubunternehmenRO subunternehmenRO;
MangelRO mangelRO;
MeldungRO meldungRO;
PlzRO plzRO;
AdresseRO adresseRO;
ObjekttypRO objekttypRO;
ArbeitstypRO arbeitstypRO;
MangelstatusRO mangelstatusRO;
MeldungstypRO meldungstypRO;
ProjektGuMitarbeiterRO projektGuMitarbeiterRO;
LoginRO loginRO;
ProjektSuMitarbeiterRO projektSuMitarbeiterRO;
BauherrRO bauherrRO;
GuMitarbeiterRO guMitarbeiterRO;
ProjektstatusRO projektstatusRO;
SuMitarbeiterRO suMitarbeiterRO;
RolleRO rolleRO;

// The Tests

/**
 * Tests if Projekte are findable
 */
@Test
public void getAllProjektTest() {
    try {
        ClientRMI client = new ClientRMI();
        projekte = client.getAllProjekt();
        assertTrue(projekte.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekte");
```

```
    }
}

/**
 * Tests if Subunternehmen are findable
 */
@Test
public void getAllSubunternehmenTest() {
    try {
        ClientRMI client = new ClientRMI();
        subunternehmen = client.getAllSubunternehmen();
        assertTrue(subunternehmen.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Subunternehmen");
    }
}

/**
 * Tests if Projekte pro Subunternehmen are findable
 */
@Test
public void getProjektproSubunternehmenTest() {
    try {
        ClientRMI client = new ClientRMI();
        anzProjekte = client.getProjektproSubunternehmen(80);
        assertNotNull(anzProjekte);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekte pro Subunternehmen");
    }
}

/**
 * Tests if Mängel are findable
 */
@Test
public void getAllMangelTest() {
    try {
        ClientRMI client = new ClientRMI();
        maengel = client.getAllMangel();
        assertTrue(maengel.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Mängel");
    }
}

/**
 * Tests if Meldungen are findable
 */
@Test
public void getAllMeldungTest() {
    try {
        ClientRMI client = new ClientRMI();
        meldung = client.getAllMeldung();
        assertTrue(meldung.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Meldung");
    }
}
```

```java
/**
 * Tests if Projekte are findable by ID
 */
@Test
public void getProjektByIdTest() {
    try {
        ClientRMI client = new ClientRMI();
        projekt = client.getProjektById(109);
        assertTrue(projekt.getId().equals(109));
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekt by ID");
    }
}

/**
 * Tests if Subunternehmen are findable by ID
 */
@Test
public void getSubunternehmenByIdTest() {
    try {
        ClientRMI client = new ClientRMI();
        subunternehmennr = client.getSubunternehmenById(74);
        assertTrue(subunternehmennr.getId().equals(74));
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Subunternehme by ID");
    }
}

/**
 * Tests if Meldungenare findable by ID
 */
@Test
public void getMeldungByIdTest() {
    try {
        ClientRMI client = new ClientRMI();
        meldungnr = client.getMeldungById(141);
        assertTrue(meldungnr.getId().equals(141));
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Meldung by ID");
    }
}

/**
 * Tests if Personen are findable
 */
@Test
public void getAllPersonTest() {
    try {
        ClientRMI client = new ClientRMI();
        person = client.getAllPerson();
        assertTrue(person.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Person");
    }
}

/**
 * Tests if Subunternehmen are updateable
 */
```

```
    @Test
    public void updateSubunternehmenTest() {
        try {
            ClientRMI client = new ClientRMI();
            client.updateSubunternehmen(client.getSubunternehmenById(74));
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update Subunternehmen");
        }
    }

    /**
     * Tests if Projekte are findable by Bezeichnung
     */
    @Test
    public void getProjektByBezeichnungTest() {
        try {
            ClientRMI client = new ClientRMI();
            projekt = client.getAllProjekt().get(1);
            projekte =
client.getProjektByBezeichnung(projekt.getBezeichnung());
            assertTrue(projekte.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Bezeichnung");
        }
    }

    /**
     * Tests if PLZ are findable
     */
    @Test
    public void getAllPlzTest() {
        try {
            ClientRMI client = new ClientRMI();
            plz = client.getAllPlz();
            assertTrue(plz.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find PLZ");
        }
    }

    /**
     * Tests if Projekte are findableby Bauherr
     */
    @Test
    public void getProjektByBauherrTest() {
        try {
            ClientRMI client = new ClientRMI();
            projekte =
client.getProjektByBauherr(client.getAllProjekt().get(0)
                    .getFkBauherr().get(0).getNachname().toString());
            assertTrue(projekte.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Bauherr");
        }
    }

    /**
     * Tests if Projekte are findable by Plz
     */
```

```
@Test
public void getProjektByPlzTest() {
    try {
        ClientRMI client = new ClientRMI();
        projekte = client.getProjektByPlz(client.getAllProjekt().get(0)
                .getFkAdresse().getPlz().getPlz().toString());
        assertTrue(projekte.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekt by Plz");
    }
}

/**
 * tests if Projekte are findable by Ort
 */
@Test
public void getProjektByOrtTest() {
    try {
        ClientRMI client = new ClientRMI();
        projekte = client.getProjektByOrt(client.getAllProjekt().get(0)
                .getFkAdresse().getPlz().getOrt());
        assertTrue(projekte.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekt by Ort");
    }
}

/**
 * tests if Projekte are findable by Objekt
 */
@Test
public void getProjektByObjekttypTest() {
    try {
        ClientRMI client = new ClientRMI();
        projekte = client.getProjektByObjekttyp(client.getAllProjekt()
                .get(2).getFkObjekttyp().getBezeichnung());
        assertTrue(projekte.size() > 0);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekt by Objekttyp");
    }
}

/**
 * tests if Projekte are findable by Arbeitstyp
 */
@Test
public void getProjektByArbeitstypTest() {
    try {
        ClientRMI client = new ClientRMI();
        projekte = client.getProjektByArbeitstyp(client.getAllProjekt()
                .get(0).getFkArbeitstyp().getBezeichnung());
        assertNotNull(projekte);
    } catch (Exception e) {
        e.printStackTrace();
        fail("Couldn't find Projekt by Arbeitstyp");
    }
}

/**
 * tests if Projekte are findable by Projektstatus
```

```
     */
    @Test
    public void getProjektByProjektstatusTest() {
        try {
            ClientRMI client = new ClientRMI();
            projekte =
client.getProjektByProjektstatus(client.getAllProjekt()
                    .get(2).getFkProjektstatus().getBezeichnung());
            assertTrue(projekte.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Projektstatus");
        }
    }

    /**
     * tests if Adressen are addable
     */
    @Test
    public void addAdresseTest() {
        try {
            ClientRMI client = new ClientRMI();
            Adresse test = new Adresse("RMI Strasse", client.getAllPlz()
                    .get(50));
            client.addAdresse(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add Adresse");
        }
    }

    /**
     * tests if Subunternehmen are addable
     */
    @Test
    public void addSubunternehmenTest() {
        try {
            ClientRMI client = new ClientRMI();

            Subunternehmen test = new Subunternehmen(client.getAllBauherr()
                    .get(3).getFkAdresse(), "RMI Unternehmen",
"0418332938");
            client.addSubunternehmen(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add Subunternehmen");
        }
    }

    /**
     * tests if Subunternehmen-Projekte are findable
     */
    @Test
    public void getAllSubunternehmenProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            suprojekte = client.getAllSubunternehmenProjekt(client
                    .getAllSubunternehmen().get(0).getId());
            assertTrue(suprojekte.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all Subunternehmenprojekte");
        }
```

```
    }

    /**
     * tests if all Mitarbeiter of a Subunternehmen are findable
     */
    @Test
    public void getAllSubunternehmenMitarbeiterTest() {
        try {
            ClientRMI client = new ClientRMI();
            sumitarbeiter = client.getAllSubunternehmenMitarbeiter(client
                    .getAllSubunternehmen().get(0));
            assertTrue(sumitarbeiter.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all Subunternehmenmitarbeiter");
        }
    }

    /**
     * tests if Objekttyp are findable
     */
    @Test
    public void getAllObjekttypTest() {
        try {
            ClientRMI client = new ClientRMI();
            objekttyp = client.getAllObjekttyp();
            assertTrue(objekttyp.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Objekttyp");
        }
    }

    /**
     * tests if Objekttyp are findable
     */
    @Test
    public void getAllArbeitstypTest() {
        try {
            ClientRMI client = new ClientRMI();
            arbeitstyp = client.getAllArbeitstyp();
            assertTrue(arbeitstyp.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find  Arbeitstyp");
        }
    }

    /**
     * tests if Mangel of a Projekt are findable
     */
    @Test
    public void getAllMangelProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            mangelOfProjekt =
client.getAllMangelProjekt(client.getAllProjekt()
                    .get(0).getId());
            assertTrue(mangelOfProjekt.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find  MangelOfProjekt");
        }
```

```
    }

    /**
     * tests if Mängel are findable by
     */
    @Test
    public void getMangelByIdTest() {
        try {
            ClientRMI client = new ClientRMI();
            mangelnr = client.getMangelById(client.getAllMangel().get(0)
                    .getId());
            assertNotNull(mangelnr);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find  MangelOfProjekt");
        }
    }

    /**
     * tests if Mängel are updateable
     */
    @Test
    public void updateMangelTest() {
        try {
            ClientRMI client = new ClientRMI();
            client.updateMangel(client.getMangelById(client.getAllMangel()
                    .get(0).getId()));
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update Mangel");
        }
    }

    /**
     * tests if Mangelstatuse are findable
     */
    @Test
    public void getAllMangelStatusTest() {
        try {
            ClientRMI client = new ClientRMI();
            mangelstatus = client.getAllMangelStatus();
            assertTrue(mangelstatus.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find  AllMangelStatus");
        }
    }

    /**
     * tests if Mängel are addable
     */
    @Test
    public void addMangelTest() {
        try {
            ClientRMI client = new ClientRMI();
            Mangel test = new Mangel(client.getAllProjekt().get(0), "RMI
Test",
                    new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(
                            2015, 06, 01),
client.getAllMangelStatus().get(0),
                    client.getLoginById(15), "RMI getestet");
            client.addMangel(test);
```

```
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add Mangel");
        }
    }

    /**
     * tests if Meldungstypen are findable
     */
    @Test
    public void getAllMeldungstypTest() {
        try {
            ClientRMI client = new ClientRMI();
            meldungstyp = client.getAllMeldungstyp();
            assertTrue(meldungstyp.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all Meldungstyp");
        }
    }

    /**
     * tests if Meldungen are addable
     */
    @Test
    public void addMeldungTest() {
        try {
            ClientRMI client = new ClientRMI();
            Meldung test = new Meldung(client.getAllMangel().get(0), client
                    .getAllMeldungstyp().get(0), "RMI Test",
                    new GregorianCalendar(2015, 06, 01), false,
                    client.getLoginById(15));
            client.addMeldung(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add Meldung");
        }
    }

    /**
     * tests if Meldungen are findable by Mangel
     */
    @Test
    public void getAllMeldungByMangelTest() {
        try {
            ClientRMI client = new ClientRMI();
            meldungByMangel = client.getAllMeldungByMangel(client
                    .getAllMangel().get(3));
            assertTrue(meldungByMangel.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all Meldung by Mangel");
        }
    }

    /**
     * tests if Unternehmen are findable by Projekt
     */
    @Test
    public void getUnternehmenByProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            subunternehmen = client.getUnternehmenByProjekt(client
```

```
                .getAllProjekt().get(3).getId());
            assertTrue(subunternehmen.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Unternehmen by Projekt");
        }
    }

    /**
     * tests if Bauleiter are findable by Projekt
     */
    @Test
    public void getBauleiterByProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            bauleiter = client.getBauleiterByProjekt(client.getAllProjekt()
                    .get(3));
            assertTrue(bauleiter.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Bauleiter by Projekt");
        }
    }

    /**
     * tests if Login are findable by Name
     */
    @Test
    public void getLoginByNameTest() {
        try {
            ClientRMI client = new ClientRMI();
            login = client.getLoginByName("bauleiter");
            assertTrue(login.getId() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Login by Name");
        }
    }

    /**
     * tests if Login are findable by ID
     */
    @Test
    public void getLoginByIdTest() {
        try {
            ClientRMI client = new ClientRMI();
            loginnr = client.getLoginById(29);
            assertTrue(loginnr.getId() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Login by ID");
        }
    }

    /**
     * tests if Projekte are updateable
     */
    @Test
    public void updateProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            client.updateProjekt(client.getAllProjekt().get(3));
        } catch (Exception e) {
```

```
            e.printStackTrace();
            fail("Couldn't update Projekt");
        }
    }

    /**
     * tests if SuMitarbeiter are addable by Projekt
     */
    @Test
    public void addSuMitarbeiterByProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            ProjektSuMitarbeiter test = new ProjektSuMitarbeiter(client
                    .getAllProjekt().get(2), client
                    .getAllSubunternehmenMitarbeiter(
                            client.getAllSubunternehmen().get(0)).get(0),
                    new GregorianCalendar(2015, 4, 01), null);
            client.addSuMitarbeiterByProjekt(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add ProjektSuMitarbeiter");
        }
    }

    /**
     * tests if Bauherren are findable
     */
    @Test
    public void getAllBauherrTest() {
        try {
            ClientRMI client = new ClientRMI();
            bauherren = client.getAllBauherr();
            assertTrue(bauherren.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all Bauherr");
        }
    }

    /**
     * tests if Projekte are addable
     */
    @Test
    public void addProjektTest() {
        try {
            // Add Adresse für Projekt
            ClientRMI client = new ClientRMI();
            Adresse adressetest = new Adresse("RMI Projekt Strasse 3",
client
                    .getAllPlz().get(55));
            client.addAdresse(adressetest);

            //Add projekt
            Projekt test = new Projekt(adressetest, "RMI Test",
client.getAllBauherr(),
                    new GregorianCalendar(2015, 5, 18), new
GregorianCalendar(
                            2015, 6, 06), client.getAllObjekttyp().get(3),
                    client.getAllArbeitstyp().get(1), new
GregorianCalendar(
                            2015, 6, 06),
client.getAllProjektstatus().get(1));
            client.addProjekt(test);
```

```
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add Projekt");
        }
    }

    /**
     * tests if GuMitarbeiter are findable
     */
    @Test
    public void getAllGuMitarbeiterTest() {
        try {
            ClientRMI client = new ClientRMI();
            guMitarbeiter = client.getAllGuMitarbeiter();
            assertTrue(guMitarbeiter.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all guMitarbeiter");
        }
    }

    /**
     * tests if GuMitarbeiter are addable by Projekt
     */
    @Test
    public void addGuMitarbeiterByProjektTest() {
        try {
            ClientRMI client = new ClientRMI();
            ProjektGuMitarbeiter test = new ProjektGuMitarbeiter(client
                    .getAllProjekt().get(2), client.getAllGuMitarbeiter()
                    .get(0), new GregorianCalendar(2015, 4, 01), null);
            client.addGuMitarbeiterByProjekt(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add GuMitarbeiter by Projekt");
        }
    }

    /**
     * tests if GuMitarbeiter are updateable
     */
    @Test
    public void updateProjektGuMitarbeiterTest() {
        try {
            ClientRMI client = new ClientRMI();
            client.updateProjektGuMitarbeiter(client.getBauleiterByProjekt(
                    client.getAllProjekt().get(0)).get(0));
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update Subunternehmen");
        }
    }

    /**
     * tests if Projektstatuse are findable
     */
    @Test
    public void getAllProjektstatusTest() {
        try {
            ClientRMI client = new ClientRMI();
            projektstatus = client.getAllProjektstatus();
            assertTrue(projektstatus.size() > 0);
        } catch (Exception e) {
```

```
            e.printStackTrace();
            fail("Couldn't find all Projektstatus");
        }
    }

    /**
     * tests if GuMitarbeiter are addable
     */
    @Test
    public void addGuMitarbeiterTest() {
        try {
            ClientRMI client = new ClientRMI();
            GuMitarbeiter test = new GuMitarbeiter("bernand", "baum",
                    "0978392929", client.getLoginById(18));
            client.addGuMitarbeiter(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add GuMitarbeiter by Projekt");
        }
    }

    /**
     * tests if SuMitarbeiter are addable
     */
    @Test
    public void addSuMitarbeiterTest() {
        try {
            ClientRMI client = new ClientRMI();
            SuMitarbeiter test = new SuMitarbeiter("RMI", "test",
"0900203090",
                    client.getSubunternehmenById(74),
client.getLoginById(18));
            client.addSuMitarbeiter(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add SuMitarbeiter");
        }
    }

    /**
     * tests if Bauherren are addable
     */
    @Test
    public void addBauherrTest() {
        try {
            ClientRMI client = new ClientRMI();
            Adresse adressetest = new Adresse("RMI Strasse 2", client
                    .getAllPlz().get(54));
            client.addAdresse(adressetest);
            Bauherr test = new Bauherr("RMI", "Bauherr", "0900393930",
                    adressetest);
            client.addBauherr(test);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't add SuMitarbeiter");
        }
    }

    /**
     * tests if Rollen are findable
     */
    @Test
    public void getAllRolleTest() {
```

```
        try {
            ClientRMI client = new ClientRMI();
            rollen = client.getAllRolle();
            assertTrue(rollen.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find all Rollen");
        }
    }

    /**
     * tests if Personen are findable by ID
     */
    @Test
    public void getPersonByIdTest() {
        try {
            ClientRMI client = new ClientRMI();
            personnr = client.getPersonById(client.getAllProjekt().get(1)
                    .getFkBauherr().get(0).getId());
            assertNotNull(personnr);

        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Person by ID");
        }
    }

    /**
     * tests if Projekte are findable by Person
     */
    @Test
    public void getProjektbyPersonTest() {
        try {
            ClientRMI client = new ClientRMI();
            projekte =
client.getProjektbyPerson(client.getAllProjekt().get(1)
                    .getFkBauherr().get(0));
            assertTrue(projekte.size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Person");
        }
    }

    /**
     * tests if Personen are updateable
     */
    @Test
    public void updatePersonTest() {
        try {
            ClientRMI client = new ClientRMI();
            client.updatePerson(client.getAllGuMitarbeiter().get(0));
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update Person");
        }
    }

    /**
     * tests if Meldungen are updateable
     */
    @Test
    public void updateMeldungTest() {
```

```
        try {
            ClientRMI client = new ClientRMI();
            client.updateMeldung(client.getAllMeldung().get(0));
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update Meldung");
        }
    }
}
```

TemporalDateTyp

```
package ch.hsluw.mangelmanager.helper;

public enum TemporalDateTyp {
        DATE, //java.sql.Date
        TIME, //java.sql.Time
        TIMESTAMP //java.sql.Timestamp
}
```

Adresse

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

/**
 * Diese Klasse bildet eine Adresse ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
public class Adresse implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String strasse;
    @ManyToOne
    private Plz plz;

    public Adresse() {

    }

    /**
     * @param strasse
```

```
     * @param plz
     *
     */
    public Adresse(String strasse, Plz plz) {
        this.strasse = strasse;
        this.plz = plz;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id
     *            the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the strasse
     */
    public String getStrasse() {
        return strasse;
    }

    /**
     * @param strasse
     *            the strasse to set
     */
    public void setStrasse(String strasse) {
        this.strasse = strasse;
    }

    /**
     * @return the plz
     */
    public Plz getPlz() {
        return plz;
    }

    /**
     * @param plz
     *            the plz to set
     */
    public void setPlz(Plz plz) {
        this.plz = plz;
    }
}
```

Arbeitstyp

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;
```

```java
import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;

/**
 * Diese Klasse bildet ein Arbeitstyp ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
@NamedQueries({
    @NamedQuery(name = "Arbeitstyp.findByBezeichnung", query = "SELECT a
FROM Arbeitstyp a WHERE a.bezeichnung=:bezeichnung")})
public class Arbeitstyp implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String bezeichnung;

    public Arbeitstyp() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @param bezeichnung
     */
    public Arbeitstyp(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id
     *            the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
```

```
     * @param bezeichnung
     *            the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

     @Override
     public String toString() {
         return bezeichnung;
     }

}
```

Bauherr

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.OneToOne;

/**
 * Diese Klasse bildet ein Bauherr ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
public class Bauherr extends Person implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

//  @Id
//  @GeneratedValue
//  private Integer id;
    @OneToOne (cascade = CascadeType.ALL)
    private Adresse fkAdresse;

    public Bauherr() {
        // TODO Auto-generated constructor stub
        super();
    }

    /**
     *
     * @param nachname
     * @param vorname
     * @param telefon
     * @param fkAdresse
     */
    public Bauherr(String nachname, String vorname, String telefon,
            Adresse fkAdresse) {
        super(nachname, vorname, telefon);
        this.fkAdresse = fkAdresse;
```

```
    }

//  /**
//   * @return the id
//   */
//  public Integer getId() {
//      return id;
//  }
//
//  /**
//   * @param id the id to set
//   */
//  public void setId(Integer id) {
//      this.id = id;
//  }

    /**
     * @return the fkAdresse
     */
    public Adresse getFkAdresse() {
        return fkAdresse;
    }

    /**
     * @param fkAdresse the fkAdresse to set
     */
    public void setFkAdresse(Adresse fkAdresse) {
        this.fkAdresse = fkAdresse;
    }

    @Override
    public String toString() {
        return getNachname() + " "+ getVorname();
    }



}
```

GuMitarbeiter

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.ManyToOne;

/**
 * Diese Klasse bildet einen SuMitarbeiter ab.
 *
 * @version 1.0
 * @author lkuendig
 *
 */

@Entity
public class GuMitarbeiter extends Person implements Serializable {
```

```
    private static final long serialVersionUID = 269913975468169 60L;

    @ManyToOne (cascade = CascadeType.ALL)
    private Login fkLogin;

    public GuMitarbeiter(){
        super();
    }

    /**
     * @param nachname
     * @param vorname
     * @param telefon
     * @param fkLogin
     *
     */

    public GuMitarbeiter(String nachname, String vorname, String telefon,
Login fkLogin){
        super(nachname, vorname, telefon);
        this.fkLogin = fkLogin;
    }

//  /**
//   * @return the id
//   */
//  public Integer getId() {
//      return id;
//  }
//
//  /**
//   * @param id the id to set
//   */
//  public void setId(Integer id) {
//      this.id = id;
//  }

    /**
     * @return the fkLogin
     */
    public Login getFkLogin() {
        return fkLogin;
    }

    /**
     * @param fkLogin the fkLogin to set
     */
    public void setFkLogin(Login fkLogin) {
        this.fkLogin = fkLogin;
    }

    @Override
    public String toString() {
        return getNachname() + getVorname();
    }


}
```

Login

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;

/**
 * Diese Klasse dient als Login.
 *
 * @version 1.0
 * @author miten
 *
 */

@Entity
@NamedQueries({
    @NamedQuery(name = "Login.findByName", query = "SELECT l FROM Login l
WHERE l.benutzername=:loginName")})

public class Login implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;

    private String benutzername;
    private String passwort;
    private String email;
    @ManyToOne
    private Rolle fkrolle;

    public Login() {

    }

    /**
     * @param benutzername
     * @param passwort
     * @param email
     * @param fkrolle
     */
    public Login(String benutzername, String passwort, String email,
            Rolle fkrolle) {
        super();
        this.benutzername = benutzername;
        this.passwort = passwort;
        this.email = email;
        this.fkrolle = fkrolle;
    }
```

```java
/**
 * @return the id
 */
public Integer getId() {
    return id;
}

/**
 * @param id the id to set
 */
public void setId(Integer id) {
    this.id = id;
}

/**
 * @return the benutzername
 */
public String getBenutzername() {
    return benutzername;
}

/**
 * @param benutzername the benutzername to set
 */
public void setBenutzername(String benutzername) {
    this.benutzername = benutzername;
}

/**
 * @return the passwort
 */
public String getPasswort() {
    return passwort;
}

/**
 * @param passwort the passwort to set
 */
public void setPasswort(String passwort) {
    this.passwort = passwort;
}

/**
 * @return the email
 */
public String getEmail() {
    return email;
}

/**
 * @param email the email to set
 */
public void setEmail(String email) {
    this.email = email;
}

/**
 * @return the fkrolle
 */
public Rolle getFkrolle() {
    return fkrolle;
}
```

```
    /**
     * @param fkrolle the fkrolle to set
     */
    public void setFkrolle(Rolle fkrolle) {
        this.fkrolle = fkrolle;
    }

}
```

Mangel

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;
import java.util.GregorianCalendar;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 * Diese Klasse bildet einen Mangel ab.
 *
 * @version 1.0
 * @author mmont
 *
 */

@Entity
public class Mangel implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    @ManyToOne
    private Projekt fkProjekt;
    private String bezeichnung;
    private String beschreibung;
    @Temporal(TemporalType.TIMESTAMP)
    private GregorianCalendar erfassungsZeit;
    @Temporal(TemporalType.TIMESTAMP)
    private GregorianCalendar abschlussZeit;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar faelligkeitsDatum;
    @ManyToOne
    private Mangelstatus fkMangelstatus;
    @ManyToOne
    private Login fkLogin;


    public Mangel() {
        // TODO Auto-generated constructor stub
    }
```

```
    /**
     * Constructor
     */
    public Mangel(Projekt fkProjekt, String bezeichnung,
            GregorianCalendar erfassungsZeit,
            GregorianCalendar faelligkeitsDatum, Mangelstatus
fkMangelstatus,
            Login fkLogin, String beschreibung) {
        super();
        this.fkProjekt = fkProjekt;
        this.bezeichnung = bezeichnung;
        this.erfassungsZeit = erfassungsZeit;
        this.abschlussZeit = abschlussZeit;
        this.faelligkeitsDatum = faelligkeitsDatum;
        this.fkMangelstatus = fkMangelstatus;
        this.fkLogin = fkLogin;
        this.beschreibung = beschreibung;
    }

    // Getters and Setters
    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id
     *            the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the fkProjekt
     */
    public Projekt getFkProjekt() {
        return fkProjekt;
    }

    /**
     * @param fkProjekt
     *            the fkProjekt to set
     */
    public void setFkProjekt(Projekt fkProjekt) {
        this.fkProjekt = fkProjekt;
    }

    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
     * @param bezeichnung
     *            the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
```

```
}

/**
 * @return the erfassungsZeit
 */
public GregorianCalendar getErfassungsZeit() {
    return erfassungsZeit;
}

/**
 * @param erfassungsZeit
 *            the erfassungsZeit to set
 */
public void setErfassungsZeit(GregorianCalendar erfassungsZeit) {
    this.erfassungsZeit = erfassungsZeit;
}

/**
 * @return the abschlussZeit
 */
public GregorianCalendar getAbschlussZeit() {
    return abschlussZeit;
}

/**
 * @param abschlussZeit
 *            the abschlussZeit to set
 */
public void setAbschlussZeit(GregorianCalendar abschlussZeit) {
    this.abschlussZeit = abschlussZeit;
}

/**
 * @return the faelligkeitsDatum
 */
public GregorianCalendar getFaelligkeitsDatum() {
    return faelligkeitsDatum;
}

/**
 * @param faelligkeitsDatum
 *            the faelligkeitsDatum to set
 */
public void setFaelligkeitsDatum(GregorianCalendar faelligkeitsDatum) {
    this.faelligkeitsDatum = faelligkeitsDatum;
}

/**
 * @return the fkMangelstatus
 */
public Mangelstatus getFkMangelstatus() {
    return fkMangelstatus;
}

/**
 * @param fkMangelstatus
 *            the fkMangelstatus to set
 */
public void setFkMangelstatus(Mangelstatus fkMangelstatus) {
    this.fkMangelstatus = fkMangelstatus;
}

/**
```

```
     * @return the fkLogin
     */
    public Login getFkLogin() {
        return fkLogin;
    }

    /**
     * @param fkLogin
     *            the fkLogin to set
     */
    public void setFkLogin(Login fkLogin) {
        this.fkLogin = fkLogin;
    }

    /**
     * @return the beschreibung
     */
    public String getBeschreibung() {
        return beschreibung;
    }

    /**
     * @param beschreibung
     *            the beschreibung to set
     */
    public void setBeschreibung(String beschreibung) {
        this.beschreibung = beschreibung;
    }
    @Override
    public String toString() {
        return id +" - " +  bezeichnung;
    }



}
```

Mangelstatus

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 * Diese Klasse bildet einen Mangelstatus ab.
 *
 * @version 1.0
 * @author mmont
 *
 */

@Entity
public class Mangelstatus implements Serializable {
```

```
    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String bezeichnung;

    public Mangelstatus(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    public Mangelstatus() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
     * @param bezeichnung the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    @Override
    public String toString() {
        return bezeichnung;
    }

}
```

Meldung

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.model;

import java.io.Serializable;
import java.util.GregorianCalendar;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
```

```java
/**
 * Diese Klasse bildet eine Meldung ab.
 *
 * @version 1.0
 * @author cdemir
 *
 */

@Entity
@NamedQueries({
@NamedQuery(name = "Meldung.findAllMeldungByMangel", query = "SELECT m FROM
Meldung m WHERE m.fkMangel=:mangelId and m.quittiert=false")})
public class Meldung implements Serializable {

    private static final long serialVersionUID = 2067831453127875781L;

    @Id
    @GeneratedValue
    private Integer id;


    private Mangel fkMangel;

    @ManyToOne
    private Meldungstyp fkMeldungstyp;
    private String text;

    @Temporal(TemporalType.DATE)
    private GregorianCalendar zeitpunkt;

    private boolean quittiert;
    private Login fkLogin;

    public Meldung() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @param fkMangel
     * @param fkMeldungstyp
     * @param text
     * @param zeitpunkt
     * @param quittiert
     * @param fkLogin
     */
    public Meldung(Mangel fkMangel, Meldungstyp fkMeldungstyp, String text,
GregorianCalendar zeitpunkt, boolean quittiert,
            Login fkLogin) {
        super();
        this.fkMangel = fkMangel;
        this.fkMeldungstyp = fkMeldungstyp;
        this.text = text;
        this.zeitpunkt = zeitpunkt;
        this.quittiert = quittiert;
        this.fkLogin = fkLogin;
    }

    /**
     * @return the id
     */
    public Integer getId() {
```

```
    return id;
}

/**
 * @param id
 *            the id to set
 */
public void setId(Integer id) {
    this.id = id;
}

/**
 * @return the fkMangel
 */
public Mangel getFkMangel() {
    return fkMangel;
}

/**
 * @param fkMangel
 *            the fkMangel to set
 */
public void setFkMangel(Mangel fkMangel) {
    this.fkMangel = fkMangel;
}

/**
 * @return the fkMeldungstyp
 */
public Meldungstyp getFkMeldungstyp() {
    return fkMeldungstyp;
}

/**
 * @param fkMeldungstyp
 *            the fkMeldungstyp to set
 */
public void setFkMeldungstyp(Meldungstyp fkMeldungstyp) {
    this.fkMeldungstyp = fkMeldungstyp;
}

/**
 * @return the text
 */
public String getText() {
    return text;
}

/**
 * @param text
 *            the text to set
 */
public void setText(String text) {
    this.text = text;
}

/**
 * @return the quittiert
 */
public boolean getQuittiert() {
    return quittiert;
}
```

```
    /**
     * @param quittiert
     *            the quittiert to set
     */
    public void setQuittiert(boolean quittiert) {
        this.quittiert = quittiert;
    }

    /**
     * @return the fkLogin
     */
    public Login getFkLogin() {
        return fkLogin;
    }

    /**
     * @param fkLogin
     *            the fkLogin to set
     */
    public void setFkLogin(Login fkLogin) {
        this.fkLogin = fkLogin;
    }

    /**
     * @return the zeitpunkt
     */
    public GregorianCalendar getZeitpunkt() {
        return zeitpunkt;
    }

    /**
     * @param zeitpunkt the zeitpunkt to set
     */
    public void setZeitpunkt(GregorianCalendar zeitpunkt) {
        this.zeitpunkt = zeitpunkt;
    }

    @Override
    public String toString() {
        return text;
    }



}
```

Meldungstyp

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 * Diese Klasse bildet einen Meldungstyp ab.
```

```
 *
 * @version 1.0
 * @author cdemir
 *
 */

@Entity
public class Meldungstyp implements Serializable {

    private static final long serialVersionUID = 6358290611507121648L;

    @Id
    @GeneratedValue
    private Integer id;
    private String bezeichnung;

    public Meldungstyp() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @param bezeichnung
     */
    public Meldungstyp(String bezeichnung) {
        super();
        this.bezeichnung = bezeichnung;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id
     *            the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
     * @param bezeichnung
     *            the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    @Override
    public String toString() {
        return bezeichnung;
    }
```

```
}
```

Objekttyp

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;

/**
 * Diese Klasse bildet ein Objekttyp ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
@NamedQueries({ @NamedQuery(name = "Objekttyp.findByBezeichnung", query =
"SELECT o FROM Objekttyp o WHERE o.bezeichnung=:bezeichnung") })
public class Objekttyp implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String bezeichnung;

    public Objekttyp() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @param bezeichnung
     */
    public Objekttyp(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id
     *            the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }
```

```java
    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
     * @param bezeichnung
     *            the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

     @Override
     public String toString() {
         return bezeichnung;
     }

}
```

Person

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.MappedSuperclass;

/**
 * Diese Klasse bildet eine Person ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@MappedSuperclass
public
abstract class Person implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String nachname;
    private String vorname;
    private String telefon;
```

```java
public Person() {
    // TODO Auto-generated constructor stub
}

/**
 * @param nachname
 * @param vorname
 * @param telefon
 */
public Person(String nachname, String vorname, String telefon) {
    this.nachname = nachname;
    this.vorname = vorname;
    this.telefon = telefon;
}

/**
 * @return the id
 */
public Integer getId() {
    return id;
}

/**
 * @param id the id to set
 */
public void setId(Integer id) {
    this.id = id;
}

/**
 * @return the nachname
 */
public String getNachname() {
    return nachname;
}

/**
 * @param nachname
 *            the nachname to set
 */
public void setNachname(String nachname) {
    this.nachname = nachname;
}

/**
 * @return the vorname
 */
public String getVorname() {
    return vorname;
}

/**
 * @param vorname
 *            the vorname to set
 */
public void setVorname(String vorname) {
    this.vorname = vorname;
}

/**
 * @return the telefon
 */
public String getTelefon() {
```

```
        return telefon;
    }

    /**
     * @param telefon
     *            the telefon to set
     */
    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }

}
```

Plz

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.Id;

/**
 * Diese Klasse bildet eine Plz ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
public class Plz implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    private Integer plz;
    @Id
    private String ort;

    public Plz() {

    }

    /**
     * @param plz
     * @param ort
     *
     */
    public Plz(Integer plz, String ort) {
        this.plz = plz;
        this.ort = ort;
    }

    /**
     * @return the plz
     */
    public Integer getPlz() {
        return plz;
```

```
    }

    /**
     * @param plz
     *            the plz to set
     */
    public void setPlz(Integer plz) {
        this.plz = plz;
    }

    /**
     * @return the ort
     */
    public String getOrt() {
        return ort;
    }

    /**
     * @param ort
     *            the ort to set
     */
    public void setOrt(String ort) {
        this.ort = ort;
    }

    @Override
    public String toString() {
        return String.valueOf(plz);
    }
}
```

Projekt

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;
import java.util.GregorianCalendar;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 * Diese Klasse bildet ein Projekt ab.
 *
 * @version 1.0
 * @author sritz
 *
```

```
 */

@Entity
@NamedQueries({
        @NamedQuery(name = "Projekt.findByBezeichnung", query = "SELECT p
FROM Projekt p WHERE p.bezeichnung LIKE :bezeichnung"),
        //@NamedQuery(name = "Projekt.findByDatumFromTillEnd", query =
"SELECT p FROM Projekt p WHERE p.startdatum >=:startdatum and p.enddatum
<=:enddatum"),
        @NamedQuery(name = "Projekt.findByBauherr", query = "SELECT p FROM
Projekt p join p.fkBauherr b where b.nachname LIKE (:bauherr) or b.vorname
LIKE (:bauherr)"),
        @NamedQuery(name = "Projekt.findByPlz", query = "SELECT p FROM
Projekt p WHERE p.fkAdresse.plz.plz =:plz"),
        @NamedQuery(name = "Projekt.findByOrt", query = "SELECT p FROM
Projekt p WHERE p.fkAdresse.plz.ort LIKE :ort"),
        @NamedQuery(name = "Projekt.findByObjekttyp", query = "SELECT p
FROM Projekt p WHERE p.fkObjekttyp.bezeichnung LIKE :objekttyp"),
        @NamedQuery(name = "Projekt.findByArbeitstyp", query = "SELECT p
FROM Projekt p WHERE p.fkArbeitstyp.bezeichnung LIKE :arbeitstyp"),
        @NamedQuery(name = "Projekt.findByProjektstatus", query = "SELECT p
FROM Projekt p WHERE p.fkProjektstatus.bezeichnung LIKE :projektstatus"),
        })
public class Projekt implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;

    @OneToOne(cascade = CascadeType.ALL)
    private Adresse fkAdresse;
    private String bezeichnung;
    private String beschreibung;

    @ManyToMany(fetch = FetchType.EAGER)
    private List<Bauherr> fkBauherr;

    @Temporal(TemporalType.DATE)
    private GregorianCalendar startDatum;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar endDatum;

    @ManyToOne
    private Objekttyp fkObjekttyp;

    @ManyToOne
    private Arbeitstyp fkArbeitstyp;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar faelligkeitsDatum;

    @ManyToOne
    private Projektstatus fkProjektstatus;

    public Projekt() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @param fkAdresse
     * @param bezeichnung
     * @param fkBauherr
```

```
     * @param startDatum
     * @param endDatum
     * @param fkObjekttyp
     * @param fkArbeitstyp
     * @param faelligkeitsDatum
     * @param fkProjektstatus
     */
    public Projekt(Adresse fkAdresse, String bezeichnung,
            List<Bauherr> fkBauherr, GregorianCalendar startDatum,
            GregorianCalendar endDatum, Objekttyp fkObjekttyp,
            Arbeitstyp fkArbeitstyp, GregorianCalendar faelligkeitsDatum,
            Projektstatus fkProjektstatus) {
        super();
        this.fkAdresse = fkAdresse;
        this.bezeichnung = bezeichnung;
        this.fkBauherr = fkBauherr;
        this.startDatum = startDatum;
        this.endDatum = endDatum;
        this.fkObjekttyp = fkObjekttyp;
        this.fkArbeitstyp = fkArbeitstyp;
        this.faelligkeitsDatum = faelligkeitsDatum;
        this.fkProjektstatus = fkProjektstatus;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the fkAdresse
     */
    public Adresse getFkAdresse() {
        return fkAdresse;
    }

    /**
     * @param fkAdresse the fkAdresse to set
     */
    public void setFkAdresse(Adresse fkAdresse) {
        this.fkAdresse = fkAdresse;
    }

    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
     * @param bezeichnung the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
```

```
        this.bezeichnung = bezeichnung;
    }

    /**
     * @return the fkBauherr
     */
    public List<Bauherr> getFkBauherr() {
        return fkBauherr;
    }

    /**
     * @param fkBauherr the fkBauherr to set
     */
    public void setFkBauherr(List<Bauherr> fkBauherr) {
        this.fkBauherr = fkBauherr;
    }

    /**
     * @return the startDatum
     */
    public GregorianCalendar getStartDatum() {
        return startDatum;
    }

    /**
     * @param startDatum the startDatum to set
     */
    public void setStartDatum(GregorianCalendar startDatum) {
        this.startDatum = startDatum;
    }

    /**
     * @return the endDatum
     */
    public GregorianCalendar getEndDatum() {
        return endDatum;
    }

    /**
     * @param endDatum the endDatum to set
     */
    public void setEndDatum(GregorianCalendar endDatum) {
        this.endDatum = endDatum;
    }

    /**
     * @return the fkObjekttyp
     */
    public Objekttyp getFkObjekttyp() {
        return fkObjekttyp;
    }

    /**
     * @param fkObjekttyp the fkObjekttyp to set
     */
    public void setFkObjekttyp(Objekttyp fkObjekttyp) {
        this.fkObjekttyp = fkObjekttyp;
    }

    /**
     * @return the fkArbeitstyp
     */
    public Arbeitstyp getFkArbeitstyp() {
```

```
        return fkArbeitstyp;
    }

    /**
     * @param fkArbeitstyp the fkArbeitstyp to set
     */
    public void setFkArbeitstyp(Arbeitstyp fkArbeitstyp) {
        this.fkArbeitstyp = fkArbeitstyp;
    }

    /**
     * @return the faelligkeitsDatum
     */
    public GregorianCalendar getFaelligkeitsDatum() {
        return faelligkeitsDatum;
    }

    /**
     * @param faelligkeitsDatum the faelligkeitsDatum to set
     */
    public void setFaelligkeitsDatum(GregorianCalendar faelligkeitsDatum) {
        this.faelligkeitsDatum = faelligkeitsDatum;
    }

    /**
     * @return the fkProjektstatus
     */
    public Projektstatus getFkProjektstatus() {
        return fkProjektstatus;
    }

    /**
     * @param fkProjektstatus the fkProjektstatus to set
     */
    public void setFkProjektstatus(Projektstatus fkProjektstatus) {
        this.fkProjektstatus = fkProjektstatus;
    }

    /**
     * @return the beschreibung
     */
    public String getBeschreibung() {
        return beschreibung;
    }

    /**
     * @param beschreibung the beschreibung to set
     */
    public void setBeschreibung(String beschreibung) {
        this.beschreibung = beschreibung;
    }


}



ProjektGuMitarbeiter
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;
import java.util.GregorianCalendar;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 * Diese Klasse bildet eine Beziehung zwischen GuMitarbeiter und Projekt
ab.
 *
 * @version 1.0
 * @author lkuendig
 *
 */

@Entity
@NamedQueries({
    @NamedQuery(name = "ProjektGuMitarbeiter.findAllBauleiterByProjekt",
query = "SELECT p FROM ProjektGuMitarbeiter p WHERE p.fkProjekt
=:projektId")})
public class ProjektGuMitarbeiter implements Serializable {

    private static final long serialVersionUID = 5806477597388591398L;

    @Id
    @GeneratedValue
    private int id;
    @ManyToOne
    private Projekt fkProjekt;
    @ManyToOne
    private GuMitarbeiter fkMitarbeiter;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar startDatum;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar endDatum;


    public ProjektGuMitarbeiter() {

    }

    /**
     * @param fkProjekt
     * @param fkuMItarbeiter
     * @param startDatum
     * @param endDatum
     *
     */

    public ProjektGuMitarbeiter(Projekt fkProjekt,GuMitarbeiter
fkMitarbeiter, GregorianCalendar startDatum,
            GregorianCalendar endDatum) {
        super();
        this.fkProjekt = fkProjekt;
        this.fkMitarbeiter = fkMitarbeiter;
```

```
        this.startDatum = startDatum;
        this.endDatum = endDatum;
    }

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(int id) {
        this.id = id;
    }

    /**
     * @return the fkProjekt
     */
    public Projekt getFkProjekt() {
        return fkProjekt;
    }

    /**
     * @param fkProjekt the fkProjekt to set
     */
    public void setFkProjekt(Projekt fkProjekt) {
        this.fkProjekt = fkProjekt;
    }

    /**
     * @return the fkMitarbeiter
     */
    public GuMitarbeiter getFkMitarbeiter() {
        return fkMitarbeiter;
    }

    /**
     * @param fkMitarbeiter the fkMitarbeiter to set
     */
    public void setFkMitarbeiter(GuMitarbeiter fkMitarbeiter) {
        this.fkMitarbeiter = fkMitarbeiter;
    }

    /**
     * @return the startDatum
     */
    public GregorianCalendar getStartDatum() {
        return startDatum;
    }

    /**
     * @param startDatum the startDatum to set
     */
    public void setStartDatum(GregorianCalendar startDatum) {
        this.startDatum = startDatum;
    }

    /**
     * @return the endDatum
     */
```

```
    public GregorianCalendar getEndDatum() {
        return endDatum;
    }

    /**
     * @param endDatum the endDatum to set
     */
    public void setEndDatum(GregorianCalendar endDatum) {
        this.endDatum = endDatum;
    }


}
```

Projektstatus

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 * Diese Klasse bildet ein Projektstatus ab.
 *
 * @version 1.0
 * @author sritz
 *
 */

@Entity
public class Projektstatus implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String bezeichnung;

    public Projektstatus() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @param bezeichnung
     */
    public Projektstatus(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
```

```
    }

    /**
     * @param id the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the bezeichnung
     */
    public String getBezeichnung() {
        return bezeichnung;
    }

    /**
     * @param bezeichnung the bezeichnung to set
     */
    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    @Override
    public String toString() {
        return bezeichnung;
    }




}
```

ProjektSuMitarbeiter

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;
import java.util.GregorianCalendar;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 * Diese Klasse bildet eine Beziehung zwischen SuMitarbeiter und Projekt
ab.
 *
 * @version 1.0
 * @author lkuendig
 *
 */

@Entity
```

```
public class ProjektSuMitarbeiter implements Serializable {

    private static final long serialVersionUID = 8347794435492517717L;

    @Id
    @GeneratedValue
    private int id;
    @ManyToOne
    private Projekt fkProjekt;
    @ManyToOne
    private SuMitarbeiter fkMitarbeiter;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar startDatum;
    @Temporal(TemporalType.DATE)
    private GregorianCalendar endDatum;

    public ProjektSuMitarbeiter(){

    }

    /**
     * @param idProjekt
     * @param idSuMItarbeiter
     * @param startDatum
     * @param endDatum
     *
     */

    public ProjektSuMitarbeiter(Projekt fkProjekt, SuMitarbeiter
fkMitarbeiter, GregorianCalendar startDatum, GregorianCalendar endDatum){
        this.fkProjekt = fkProjekt;
        this.fkMitarbeiter = fkMitarbeiter;
        this.startDatum = startDatum;
        this.endDatum = endDatum;
    }

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(int id) {
        this.id = id;
    }

    /**
     * @return the fkProjekt
     */
    public Projekt getFkProjekt() {
        return fkProjekt;
    }

    /**
     * @param fkProjekt the fkProjekt to set
     */
    public void setFkProjekt(Projekt fkProjekt) {
        this.fkProjekt = fkProjekt;
    }
```

```
    /**
     * @return the fkMitarbeiter
     */
    public SuMitarbeiter getFkMitarbeiter() {
        return fkMitarbeiter;
    }

    /**
     * @param fkMitarbeiter the fkMitarbeiter to set
     */
    public void setFkMitarbeiter(SuMitarbeiter fkMitarbeiter) {
        this.fkMitarbeiter = fkMitarbeiter;
    }

    /**
     * @return the startDatum
     */
    public GregorianCalendar getStartDatum() {
        return startDatum;
    }

    /**
     * @param startDatum the startDatum to set
     */
    public void setStartDatum(GregorianCalendar startDatum) {
        this.startDatum = startDatum;
    }

    /**
     * @return the endDatum
     */
    public GregorianCalendar getEndDatum() {
        return endDatum;
    }

    /**
     * @param endDatum the endDatum to set
     */
    public void setEndDatum(GregorianCalendar endDatum) {
        this.endDatum = endDatum;
    }

     @Override
     public String toString() {
         return fkMitarbeiter.getFkSubunternehmen().getName();
     }


}
```

Rolle

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.Entity;
```

```java
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 * Diese Klasse dient als Rolle.
 *
 * @version 1.0
 * @author miten
 *
 */

@Entity
public class Rolle implements Serializable {

    private static final long serialVersionUID = 6294667886934890151L;

    @Id
    @GeneratedValue
    private Integer id;
    private String name;

    public Rolle() {

    }



    /**
     * @param name
     */
    public Rolle(String name) {
        this.name = name;
    }



    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @param name the name to set
     */
    public void setName(String name) {
        this.name = name;
    }
```

```
    @Override
    public String toString() {
        return name;
    }

}
```

Subunternehmen

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;

/**
 * Diese Klasse bildet ein Subunternehmen ab.
 *
 * @version 1.0
 * @author lkuendig
 *
 */

@Entity
@NamedQueries({
    @NamedQuery(name = "Subunternehmen.findBySubunternehmenMitarbeiter",
query = "SELECT sm FROM SuMitarbeiter sm WHERE
sm.fkSubunternehmen=:subunternehmenId"),
    @NamedQuery(name = "Subunternehmen.findAllSubunternehmenByProjekt",
query = "SELECT DISTINCT sm.fkSubunternehmen FROM SuMitarbeiter sm,
ProjektSuMitarbeiter ps WHERE ps.fkMitarbeiter = sm.id AND ps.fkProjekt
=:projektId")})
public class Subunternehmen implements Serializable {

    private static final long serialVersionUID = -2526718021212938075L;

    @Id
    @GeneratedValue
    private Integer id;
    @OneToOne(cascade = CascadeType.ALL)
    private Adresse fkAdresse;
    private String name;
    private String telefon;

    public Subunternehmen() {

    }

    /**
     * @param adresse
     * @param name
```

```
     * @param telefon
     *
     */
    public Subunternehmen(Adresse fkAdresse, String name,
            String telefon) {
        super();
        this.fkAdresse = fkAdresse;
        this.name = name;
        this.telefon = telefon;
    }

    /**
     * @return the id
     */
    public Integer getId() {
        return id;
    }

    /**
     * @param id
     *            the id to set
     */
    public void setId(Integer id) {
        this.id = id;
    }

    /**
     * @return the fkAdresse
     */
    public Adresse getFkAdresse() {
        return fkAdresse;
    }

    /**
     * @param fkAdresse
     *            the fkAdresse to set
     */
    public void setFkAdresse(Adresse fkAdresse) {
        this.fkAdresse = fkAdresse;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @param name
     *            the name to set
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * @return the telefon
     */
    public String getTelefon() {
        return telefon;
    }
```

```
    /**
     * @param telefon
     *            the telefon to set
     */
    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }

//  /**
//   * @return the fkSuMitarbeiter
//   */
//  public List<SuMitarbeiter> getFkSuMitarbeiter() {
//      return fkSuMitarbeiter;
//  }
//
//  /**
//   * @param fkSuMitarbeiter the fkSuMitarbeiter to set
//   */
//  public void setFkSuMitarbeiter(List<SuMitarbeiter> fkSuMitarbeiter) {
//      this.fkSuMitarbeiter = fkSuMitarbeiter;
//  }

    public String toString() {
        return name;
    }


}
```

SuMitarbeiter

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
package ch.hsluw.mangelmanager.model;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.ManyToOne;


/**
 * Diese Klasse bildet einen SuMitarbeiter ab.
 *
 * @version 1.0
 * @author lkuendig
 *
 */

@Entity
public class SuMitarbeiter extends Person implements Serializable {

    private static final long serialVersionUID = 1751601309829678863L;

    @ManyToOne
    private Subunternehmen fkSubunternehmen;
    @ManyToOne(cascade = CascadeType.ALL)
    private Login fkLogin;
```

```
    public SuMitarbeiter() {
        super();
    }

    /**
     * @param nachname
     * @param vorname
     * @param telefon
     * @param fkSubunternehmen
     * @param fkLogin
     *
     */

    public SuMitarbeiter(String nachname, String vorname,
            String telefon, Subunternehmen fkSubunternehmen, Login fkLogin)
{
        super(nachname, vorname, telefon);
        this.fkSubunternehmen = fkSubunternehmen;
        this.fkLogin = fkLogin;
    }


    /**
     * @return the fkSubunternehmen
     */
    public Subunternehmen getFkSubunternehmen() {
        return fkSubunternehmen;
    }

    /**
     * @param fkSubunternehmen
     *            the fkSubunternehmen to set
     */
    public void setFkSubunternehmen(Subunternehmen fkSubunternehmen) {
        this.fkSubunternehmen = fkSubunternehmen;
    }

    /**
     * @return the fkLogin
     */
    public Login getFkLogin() {
        return fkLogin;
    }

    /**
     * @param fkLogin
     *            the fkLogin to set
     */
    public void setFkLogin(Login fkLogin) {
        this.fkLogin = fkLogin;
    }


    public String toString() {
        return getNachname() + " " + getVorname();
    }


}
```

AdresseDAO

```java
package ch.hsluw.mangelmanager.persister.dao.adresse;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Adresse;

/**
 * Interface fuer Adresse Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface AdresseDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Adresse entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Adresse update(Adresse entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Adresse entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteAdresseById(Integer id) throws Exception;

    /**
     * Liefert die Adresse-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Adresse findAdresseById(Integer id);

    /**
```

```
     * Liefert alle Adresse-Objekte zurück.
     *
     * @return
     */
    List<Adresse> findAllAdresse();


}
```

AdresseDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.adresse;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface fuer Adresse Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public class AdresseDAOImpl implements AdresseDAO {
    @Override
    public void save(Adresse entity) throws Exception {
        new GenericPersisterImpl<Adresse>(Adresse.class).save(entity);
    }

    @Override
    public Adresse update(Adresse entity) throws Exception {
        return new
GenericPersisterImpl<Adresse>(Adresse.class).update(entity);
    }

    @Override
    public void delete(Adresse entity) throws Exception {
        new GenericPersisterImpl<Adresse>(Adresse.class).delete(entity);
    }

    @Override
    public void deleteAdresseById(Integer id) throws Exception {
        new GenericPersisterImpl<Adresse>(Adresse.class).deleteById(id);
    }

    @Override
    public Adresse findAdresseById(Integer id) {
        return new
GenericPersisterImpl<Adresse>(Adresse.class).findById(id);
    }

    @Override
    public List<Adresse> findAllAdresse() {
        return new GenericPersisterImpl<Adresse>(Adresse.class).findAll();
    }
```

```
}
```

ArbeitstypDAO

```
package ch.hsluw.mangelmanager.persister.dao.arbeitstyp;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Arbeitstyp;

/**
 * Interface fuer Arbeitstyp Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ArbeitstypDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Arbeitstyp entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Arbeitstyp update(Arbeitstyp entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Arbeitstyp entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteArbeitstypById(Integer id) throws Exception;

    /**
     * Liefert die Arbeitstyp-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
```

```
     */
    Arbeitstyp findArbeitstypById(Integer id);

    /**
     * Liefert alle Arbeitstyp-Objekte zurück.
     *
     * @return
     */
    List<Arbeitstyp> findAllArbeitstyp();

    /**
     * Liefert die Liste mit Arbeitstypen für die übergebene Bezeichnung
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    public List<Arbeitstyp> findArbeitstypByBezeichnung(String
bezeichnung);

}
```

ArbeitstypDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.arbeitstyp;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface fuer Arbeitstyp Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ArbeitstypDAOImpl implements ArbeitstypDAO {
    @Override
    public void save(Arbeitstyp entity) throws Exception {
        new
GenericPersisterImpl<Arbeitstyp>(Arbeitstyp.class).save(entity);
    }

    @Override
    public Arbeitstyp update(Arbeitstyp entity) throws Exception {
        return new
GenericPersisterImpl<Arbeitstyp>(Arbeitstyp.class).update(entity);
    }
```

```
    @Override
    public void delete(Arbeitstyp entity) throws Exception {
        new
GenericPersisterImpl<Arbeitstyp>(Arbeitstyp.class).delete(entity);
    }

    @Override
    public void deleteArbeitstypById(Integer id) throws Exception {
        new
GenericPersisterImpl<Arbeitstyp>(Arbeitstyp.class).deleteById(id);
    }

    @Override
    public Arbeitstyp findArbeitstypById(Integer id) {
        return new
GenericPersisterImpl<Arbeitstyp>(Arbeitstyp.class).findById(id);
    }

    @Override
    public List<Arbeitstyp> findAllArbeitstyp() {
        return new
GenericPersisterImpl<Arbeitstyp>(Arbeitstyp.class).findAll();
    }

    @Override
    public List<Arbeitstyp> findArbeitstypByBezeichnung(String bezeichnung)
{

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Arbeitstyp> tQuery =
em.createNamedQuery("Arbeitstyp.findByBezeichnung",
                Arbeitstyp.class);

        tQuery.setParameter("bezeichnung", bezeichnung);

        List<Arbeitstyp> arbeitstypListe = tQuery.getResultList();

        em.close();

        return arbeitstypListe != null ? arbeitstypListe : new
ArrayList<Arbeitstyp>();
    }

}
```

BauherrDAO

```
package ch.hsluw.mangelmanager.persister.dao.bauherr;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Bauherr;

/**
 * Interface fuer Bauherr Entity
```

```
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface BauherrDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Bauherr entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Bauherr update(Bauherr entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Bauherr entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteBauherrById(Integer id) throws Exception;

    /**
     * Liefert die Bauherr-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Bauherr findBauherrById(Integer id);

    /**
     * Liefert alle Bauherr-Objekte zurück.
     *
     * @return
     */
    List<Bauherr> findAllBauherr();


}
```

BauherrDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.bauherr;
```

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface fuer Bauherr Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public class BauherrDAOImpl implements BauherrDAO {
    @Override
    public void save(Bauherr entity) throws Exception {
        new GenericPersisterImpl<Bauherr>(Bauherr.class).save(entity);
    }

    @Override
    public Bauherr update(Bauherr entity) throws Exception {
        return new
GenericPersisterImpl<Bauherr>(Bauherr.class).update(entity);
    }

    @Override
    public void delete(Bauherr entity) throws Exception {
        new GenericPersisterImpl<Bauherr>(Bauherr.class).delete(entity);
    }

    @Override
    public void deleteBauherrById(Integer id) throws Exception {
        new GenericPersisterImpl<Bauherr>(Bauherr.class).deleteById(id);
    }

    @Override
    public Bauherr findBauherrById(Integer id) {
        return new
GenericPersisterImpl<Bauherr>(Bauherr.class).findById(id);
    }

    @Override
    public List<Bauherr> findAllBauherr() {
        return new GenericPersisterImpl<Bauherr>(Bauherr.class).findAll();
    }

}
```

GuMitarbeiterDAO

```java
package ch.hsluw.mangelmanager.persister.dao.gumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;
```

```java
import ch.hsluw.mangelmanager.model.GuMitarbeiter;

/**
 * Interface fuer GuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface GuMitarbeiterDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(GuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    GuMitarbeiter update(GuMitarbeiter entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(GuMitarbeiter entity) throws Exception;

    void deleteGuMitarbeiterById(Integer id) throws Exception;

    /**
     * Liefert die GuMitarbeiter-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    GuMitarbeiter findGuMitarbeiterById(Integer id);

    /**
     * Liefert alle GuMitarbeiter-Objekte zurück.
     *
     * @return
     */
    List<GuMitarbeiter> findAllGuMitarbeiter();

}
```

GuMitarbeiterDAOImpl

```java
package ch.hsluw.mangelmanager.persister.dao.gumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
import java.util.List;

import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface implementation fuer GuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class GuMitarbeiterDAOImpl implements GuMitarbeiterDAO {
    @Override
    public void save(GuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).save(entity);
    }

    @Override
    public GuMitarbeiter update(GuMitarbeiter entity) throws Exception {
        return new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).update(entity);
    }

    @Override
    public void delete(GuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).delete(entity);
    }

    @Override
    public void deleteGuMitarbeiterById(Integer id) throws Exception {
        new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).deleteById(id);
    }

    @Override
    public GuMitarbeiter findGuMitarbeiterById(Integer id) {
        return new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).findById(id);
    }

    @Override
    public List<GuMitarbeiter> findAllGuMitarbeiter() {
        return new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).findAll();
    }
}
```

LoginDAO

```java
package ch.hsluw.mangelmanager.persister.dao.login;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
import java.util.List;

import ch.hsluw.mangelmanager.model.Login;

/**
 * Interface fuer Login Entity
 *
 * @version 1.0
 * @author miten
 *
 */
public interface LoginDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Login entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Login update(Login entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Login entity) throws Exception;

    void deleteLoginById(Integer id) throws Exception;

    /**
     * Liefert die Login-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Login findLoginById(Integer id);

    /**
     * Liefert alle Login-Objekte zurück.
     *
     * @return
     */
    List<Login> findAllLogin();

    Login findByName(String name);

}
```

LoginDAOImpl

```java
package ch.hsluw.mangelmanager.persister.dao.login;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface fuer Login Entity
 *
 * @version 1.0
 * @author miten
 *
 */
public class LoginDAOImpl implements LoginDAO {
    @Override
    public void save(Login entity) throws Exception {
        new GenericPersisterImpl<Login>(Login.class).save(entity);
    }

    @Override
    public Login update(Login entity) throws Exception {
        return new GenericPersisterImpl<Login>(Login.class).update(entity);
    }

    @Override
    public void delete(Login entity) throws Exception {
        new GenericPersisterImpl<Login>(Login.class).delete(entity);
    }

    @Override
    public void deleteLoginById(Integer id) throws Exception {
        new GenericPersisterImpl<Login>(Login.class).deleteById(id);
    }

    @Override
    public Login findLoginById(Integer id) {
        return new GenericPersisterImpl<Login>(Login.class).findById(id);
    }

    @Override
    public List<Login> findAllLogin() {
        return new GenericPersisterImpl<Login>(Login.class).findAll();
    }

    @Override
    public Login findByName(String name) {
        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Login> tQuery = em.createNamedQuery("Login.findByName",
                Login.class);
```

```
        tQuery.setParameter("loginName", name);

        try{
        Login login = tQuery.getSingleResult();
        em.close();
        return login;
        } catch(NoResultException e) {
            em.close();
            return null;
        }

    }

}
```

MangelDAO

```
package ch.hsluw.mangelmanager.persister.dao.mangel;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.Date;
import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;

/**
 * Interface fuer Mangel Entity
 *
 * @version 1.0
 * @author mmont
 *
 */
public interface MangelDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Mangel entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Mangel update(Mangel entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Mangel entity) throws Exception;
```

```
    void deleteMangelById(Integer id) throws Exception;

    /**
     * Liefert die Mangel-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Mangel findMangelById(Integer id);

    /**
     * Liefert alle Mangel-Objekte zurück.
     *
     * @return
     */
    List<Mangel> findAllMangel();

    /**
     * Liefert die Liste mit Mängeln für die übergebene Bezeichnung zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    public List<Mangel> findMangelByBezeichnung(String bezeichnung);

    /**
     * Liefert die Liste mit Mängeln für den übergebenen Namen zurück,
falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param name
     * @return
     */
    public List<Mangel> findMangelByName(String name);

    /**
     * Liefert die Liste mit Mängeln für die übergebene Erfassungszeit
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param erfassungszeit
     * @return
     */
    public List<Mangel> findMangelByErfassungszeit(Date erfassungsZeit);

    /**
     * Liefert die Liste mit Mängeln für das übergebene Faelligkeitsdatum
     * zurück, falls welche gefunden, sonst eine leere Liste.
     *
     * @param faelligkeitsDatum
     * @return
     */
    public List<Mangel> findMangelByFaelligkeitsDatum(Date
faelligkeitsDatum);

    /**
     * Liefert die Liste mit Mängeln für den übergebenen Mangelstatus
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param mangelstatus
```

```
     * @return
     */
    public List<Mangel> findMangelByMangelstatus(String mangelstatus);

    /**
     * Liefert die Liste mit Mängeln für die übergebene AbschlussZeit
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param abschlussZeit
     * @return
     */
    public List<Mangel> findMangelByAbschlussZeit(Date abschlussZeit);

    /**
     * Liefert alle Mängel vom Projekt
     * @param projekt
     * @return
     */
    public List<Mangel> findAllMangelProjekt(Integer projekt);


}
```

MangelDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.mangel;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface fuer Mangel Entity
 *
 * @version 1.0
 * @author mmont
 *
 */
public class MangelDAOImpl implements MangelDAO {
    @Override
    public void save(Mangel entity) throws Exception {
        new GenericPersisterImpl<Mangel>(Mangel.class).save(entity);
    }

    @Override
    public Mangel update(Mangel entity) throws Exception {
        return new
GenericPersisterImpl<Mangel>(Mangel.class).update(entity);
```

```
    }

    @Override
    public void delete(Mangel entity) throws Exception {
        new GenericPersisterImpl<Mangel>(Mangel.class).delete(entity);
    }

    @Override
    public void deleteMangelById(Integer id) throws Exception {
        new GenericPersisterImpl<Mangel>(Mangel.class).deleteById(id);
    }

    @Override
    public Mangel findMangelById(Integer id) {
        return new GenericPersisterImpl<Mangel>(Mangel.class).findById(id);
    }

    @Override
    public List<Mangel> findAllMangel() {
        return new GenericPersisterImpl<Mangel>(Mangel.class).findAll();
    }

    @Override
    public List<Mangel> findMangelByBezeichnung(String bezeichnung) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Mangel> tQuery = em.createNamedQuery(
                "Mangel.findByBezeichnung", Mangel.class);

        tQuery.setParameter("bezeichnung", bezeichnung);

        List<Mangel> MangelListe = tQuery.getResultList();

        em.close();

        return MangelListe != null ? MangelListe : new ArrayList<Mangel>();
    }

    @Override
    public List<Mangel> findMangelByMangelstatus(String Mangelstatus) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Mangel> tQuery = em.createNamedQuery(
                "Mangel.findByMangelstatus", Mangel.class);

        tQuery.setParameter("Mangelstatus", Mangelstatus);

        List<Mangel> MangelListe = tQuery.getResultList();

        em.close();

        return MangelListe != null ? MangelListe : new ArrayList<Mangel>();
    }

    @Override
    public List<Mangel> findMangelByName(String name) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Mangel> tQuery =
em.createNamedQuery("Mangel.findByName",
```

```
                Mangel.class);

        tQuery.setParameter("ort", name);

        List<Mangel> MangelListe = tQuery.getResultList();

        em.close();

        return MangelListe != null ? MangelListe : new ArrayList<Mangel>();
    }

    @Override
    public List<Mangel> findMangelByErfassungszeit(Date erfassungsZeit) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Mangel> tQuery = em.createNamedQuery(
                "Mangel.findByErfassungszeit", Mangel.class);

        tQuery.setParameter("erfassungsZeit", erfassungsZeit);

        List<Mangel> MangelListe = tQuery.getResultList();

        em.close();

        return MangelListe != null ? MangelListe : new ArrayList<Mangel>();
    }

    @Override
    public List<Mangel> findMangelByFaelligkeitsDatum(Date
faelligkeitsDatum) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Mangel> tQuery = em.createNamedQuery(
                "Mangel.findByFaelligkeitsDatum", Mangel.class);

        tQuery.setParameter("faelligkeitsDatum", faelligkeitsDatum);

        List<Mangel> MangelListe = tQuery.getResultList();

        em.close();

        return MangelListe != null ? MangelListe : new ArrayList<Mangel>();
    }

    @Override
    public List<Mangel> findMangelByAbschlussZeit(Date abschlussZeit) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Mangel> tQuery = em.createNamedQuery(
                "Mangel.findByabschlussZeit", Mangel.class);

        tQuery.setParameter("abschlussZeit", abschlussZeit);

        List<Mangel> MangelListe = tQuery.getResultList();

        em.close();

        return MangelListe != null ? MangelListe : new ArrayList<Mangel>();
    }
```

```
    @Override
    public List<Mangel> findAllMangelProjekt(Integer projekt) {
        EntityManager em = JpaUtil.createEntityManager();

        Query tQuery = em.createNativeQuery(
                "select distinct m.* from mangel as m, projekt as p,
mangelstatus ms "
                + "where m.fkprojekt_id = "+projekt
                + "and m.fkmangelstatus_id = ms.id "
                + "and ms.bezeichnung = 'Offen'", Mangel.class);

        List<Mangel> mangelListe = tQuery.getResultList();

        em.close();

        return mangelListe != null ? mangelListe : new ArrayList<Mangel>();
    }

}
```

MangelstatusDAO

```
package ch.hsluw.mangelmanager.persister.dao.mangelstatus;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Mangelstatus;

/**
 * Interface fuer Mangelstatus Entity
 *
 * @version 1.0
 * @author mmont
 *
 */
public interface MangelstatusDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Mangelstatus entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Mangelstatus update(Mangelstatus entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
```

```
     * @param entity
     * @throws Exception
     */
    void delete(Mangelstatus entity) throws Exception;

    void deleteMangelstatusById(Integer id) throws Exception;

    /**
     * Liefert die Mangelstatus-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Mangelstatus findMangelstatusById(Integer id);

    /**
     * Liefert alle Mangelstatus-Objekte zurück.
     *
     * @return
     */
    List<Mangelstatus> findAllMangelstatus();
}
```

MangelstatusDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.mangelstatus;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */


import java.util.List;

import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;


/**
 * Interface fuer Mangel Entity
 *
 * @version 1.0
 * @author mmont
 *
 */
public class MangelstatusDAOImpl implements MangelstatusDAO {

    @Override
    public void save(Mangelstatus entity) throws Exception {
        new
GenericPersisterImpl<Mangelstatus>(Mangelstatus.class).save(entity);

    }

    @Override
    public Mangelstatus update(Mangelstatus entity) throws Exception {
        return new GenericPersisterImpl<Mangelstatus>(Mangelstatus.class)
                .update(entity);

    }
```

```
    @Override
    public void delete(Mangelstatus entity) throws Exception {
        new GenericPersisterImpl<Mangelstatus>(Mangelstatus.class)
                .delete(entity);

    }

    @Override
    public void deleteMangelstatusById(Integer id) throws Exception {
        new GenericPersisterImpl<Mangelstatus>(Mangelstatus.class)
                .deleteById(id);
    }

    @Override
    public Mangelstatus findMangelstatusById(Integer id) {
        return new GenericPersisterImpl<Mangelstatus>(Mangelstatus.class)
                .findById(id);
    }

    @Override
    public List<Mangelstatus> findAllMangelstatus() {
        return new GenericPersisterImpl<Mangelstatus>(Mangelstatus.class)
                .findAll();

    }
}
```

MeldungDAO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.persister.dao.meldung;

import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;



/**
 * Interface fuer Meldung Entity
 *
 * @version 1.0
 * @author cdemir
 *
 */
public interface MeldungDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Meldung entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
```

```
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Meldung update(Meldung entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Meldung entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteMeldungById(Integer id) throws Exception;

    /**
     * Liefert die Meldung-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Meldung findMeldungById(Integer id);

    /**
     * Liefert alle Meldung-Meldung zurück.
     *
     * @return
     */
    List<Meldung> findAllMeldung();

    List<Meldung> findAllMeldungByMangel(Mangel mangel);

}
```

MeldungDAOImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.persister.dao.meldung;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;
```

```java
/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MeldungManager zur Verfügung.
 *
 * @version 1.0
 * @author cdemir
 *
 */
public class MeldungDAOImpl implements MeldungDAO {
    @Override
    public void save(Meldung entity) throws Exception {
        new GenericPersisterImpl<Meldung>(Meldung.class).save(entity);
    }

    @Override
    public Meldung update(Meldung entity) throws Exception {
        return new
GenericPersisterImpl<Meldung>(Meldung.class).update(entity);
    }

    @Override
    public void delete(Meldung entity) throws Exception {
        new GenericPersisterImpl<Meldung>(Meldung.class).delete(entity);
    }

    @Override
    public void deleteMeldungById(Integer id) throws Exception {
        // TODO Auto-generated method stub

    }

    @Override
    public Meldung findMeldungById(Integer id) {
        // TODO Auto-generated method stub
        return new
GenericPersisterImpl<Meldung>(Meldung.class).findById(id);
    }

    @Override
    public List<Meldung> findAllMeldung() {
        // TODO Auto-generated method stub
        return new GenericPersisterImpl<Meldung>(Meldung.class).findAll();
    }

    @Override
    public List<Meldung> findAllMeldungByMangel(Mangel mangel) {
EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Meldung> tQuery =
em.createNamedQuery("Meldung.findAllMeldungByMangel",
                Meldung.class);

        tQuery.setParameter("mangelId", mangel);

        List<Meldung> meldungListe = tQuery.getResultList();

        em.close();

        return meldungListe != null ? meldungListe : new
ArrayList<Meldung>();
    }
```

```
}
```

MeldungstypDAO

```
package ch.hsluw.mangelmanager.persister.dao.meldungstyp;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Meldungstyp;



/**
 * Implementierung fuer MeldungstypDAO
 *
 * @version 1.0
 * @author cdemir
 *
 */
public interface MeldungstypDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Meldungstyp entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Meldungstyp update(Meldungstyp entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Meldungstyp entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteMeldungstypById(Integer id) throws Exception;

    /**
     * Liefert die Meldungstyp-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
```

```
     * @return
     */
    Meldungstyp findMeldungstypById(Integer id);

    /**
     * Liefert alle Meldungstyp-Objekte zurück.
     *
     * @return
     */
    List<Meldungstyp> findAllMeldungstyp();
}
```

MeldungstypDAOImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.persister.dao.meldungstyp;

import java.util.List;

import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MeldungstypManager zur Verfügung.
 *
 * @version 1.0
 * @author cdemir
 *
 */
public class MeldungstypDAOImpl implements MeldungstypDAO {
    @Override
    public void save(Meldungstyp entity) throws Exception {
        new
GenericPersisterImpl<Meldungstyp>(Meldungstyp.class).save(entity);
    }

    @Override
    public Meldungstyp update(Meldungstyp entity) throws Exception {
        return new
GenericPersisterImpl<Meldungstyp>(Meldungstyp.class).update(entity);
    }

    @Override
    public void delete(Meldungstyp entity) throws Exception {
        new
GenericPersisterImpl<Meldungstyp>(Meldungstyp.class).delete(entity);
    }

    @Override
    public void deleteMeldungstypById(Integer id) throws Exception {
        // TODO Auto-generated method stub

    }

    @Override
```

```
    public Meldungstyp findMeldungstypById(Integer id) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public List<Meldungstyp> findAllMeldungstyp() {
        // TODO Auto-generated method stub
        return new
GenericPersisterImpl<Meldungstyp>(Meldungstyp.class).findAll();
    }
}
```

ObjekttypDAO

```
package ch.hsluw.mangelmanager.persister.dao.objekttyp;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Objekttyp;

/**
 * Interface fuer Objekttyp Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ObjekttypDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Objekttyp entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Objekttyp update(Objekttyp entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Objekttyp entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
```

```
     * @param entity
     * @throws Exception
     */
    void deleteObjekttypById(Integer id) throws Exception;

    /**
     * Liefert die Objekttyp-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Objekttyp findObjekttypById(Integer id);

    /**
     * Liefert alle Objekttyp-Objekte zurück.
     *
     * @return
     */
    List<Objekttyp> findAllObjekttyp();

    /**
     * Liefert die Liste mit Objekttypen für die übergebene Bezeichnung
zurück,
     * falls welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    public List<Objekttyp> findObjekttypByBezeichnung(String bezeichnung);

}
```

ObjekttypDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.objekttyp;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface fuer Objekttyp Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public class ObjekttypDAOImpl implements ObjekttypDAO {
    @Override
    public void save(Objekttyp entity) throws Exception {
```

```
        new GenericPersisterImpl<Objekttyp>(Objekttyp.class).save(entity);
    }

    @Override
    public Objekttyp update(Objekttyp entity) throws Exception {
        return new
GenericPersisterImpl<Objekttyp>(Objekttyp.class).update(entity);
    }

    @Override
    public void delete(Objekttyp entity) throws Exception {
        new
GenericPersisterImpl<Objekttyp>(Objekttyp.class).delete(entity);
    }

    @Override
    public void deleteObjekttypById(Integer id) throws Exception {
        new
GenericPersisterImpl<Objekttyp>(Objekttyp.class).deleteById(id);
    }

    @Override
    public Objekttyp findObjekttypById(Integer id) {
        return new
GenericPersisterImpl<Objekttyp>(Objekttyp.class).findById(id);
    }

    @Override
    public List<Objekttyp> findAllObjekttyp() {
        return new
GenericPersisterImpl<Objekttyp>(Objekttyp.class).findAll();
    }

    @Override
    public List<Objekttyp> findObjekttypByBezeichnung(String bezeichnung) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Objekttyp> tQuery =
em.createNamedQuery("Objekttyp.findByBezeichnung",
                Objekttyp.class);

        tQuery.setParameter("bezeichnung", bezeichnung);

        List<Objekttyp> arbeitstypListe = tQuery.getResultList();

        em.close();

        return arbeitstypListe != null ? arbeitstypListe : new
ArrayList<Objekttyp>();
    }

}
```

PersonDAO

```
package ch.hsluw.mangelmanager.persister.dao.person;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
import java.util.List;

import ch.hsluw.mangelmanager.model.Person;

/**
 * Interface fuer Person Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface PersonDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Person entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Person update(Person entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Person entity) throws Exception;

    void deletePersonById(Integer id) throws Exception;

    /**
     * Liefert die Person-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Person findPersonById(Integer id);

    /**
     * Liefert alle Person-Objekte zurück.
     *
     * @return
     */
    List<Person> findAllPerson();

}
```

PersonDAOImpl

```java
package ch.hsluw.mangelmanager.persister.dao.person;
```

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface fuer Person Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public class PersonDAOImpl implements PersonDAO {
    @Override
    public void save(Person entity) throws Exception {
        new GenericPersisterImpl<Person>(Person.class).save(entity);
    }

    @Override
    public Person update(Person entity) throws Exception {
        if(entity instanceof SuMitarbeiter){
            return new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).update((SuMitarbei
ter) entity);
        }
        else if(entity instanceof GuMitarbeiter){
            return new
GenericPersisterImpl<GuMitarbeiter>(GuMitarbeiter.class).update((GuMitarbei
ter)entity);
        }else{
            return new
GenericPersisterImpl<Bauherr>(Bauherr.class).update((Bauherr)entity);
        }
    }

    @Override
    public void delete(Person entity) throws Exception {
        new GenericPersisterImpl<Person>(Person.class).delete(entity);
    }

    @Override
    public void deletePersonById(Integer id) throws Exception {
        new GenericPersisterImpl<Person>(Person.class).deleteById(id);
    }

    @Override
    public Person findPersonById(Integer id) {
        return new GenericPersisterImpl<Person>(Person.class).findById(id);
    }

    @Override
    public List<Person> findAllPerson() {
        return new GenericPersisterImpl<Person>(Person.class).findAll();
```

```
    }

}
```

PlzDAO

```
package ch.hsluw.mangelmanager.persister.dao.plz;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Plz;

/**
 * Interface fuer Plz Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface PlzDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Plz entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Plz update(Plz entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Plz entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deletePlzById(Integer id) throws Exception;

    /**
     * Liefert die Plz-Entity für den übergebenen Id-Wert zurück.
     *
```

```
     * @param id
     * @return
     */
    Plz findPlzById(Integer id);

    /**
     * Liefert alle Plz-Objekte zurück.
     *
     * @return
     */
    List<Plz> findAllPlz();


}
```

PlzDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.plz;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface fuer Plz Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public class PlzDAOImpl implements PlzDAO {
    @Override
    public void save(Plz entity) throws Exception {
        new GenericPersisterImpl<Plz>(Plz.class).save(entity);
    }

    @Override
    public Plz update(Plz entity) throws Exception {
        return new GenericPersisterImpl<Plz>(Plz.class).update(entity);
    }

    @Override
    public void delete(Plz entity) throws Exception {
        new GenericPersisterImpl<Plz>(Plz.class).delete(entity);
    }

    @Override
    public void deletePlzById(Integer id) throws Exception {
        new GenericPersisterImpl<Plz>(Plz.class).deleteById(id);
    }

    @Override
    public Plz findPlzById(Integer id) {
        return new GenericPersisterImpl<Plz>(Plz.class).findById(id);
    }
```

```
    @Override
    public List<Plz> findAllPlz() {
        return new GenericPersisterImpl<Plz>(Plz.class).findAll();
    }

}
```

ProjektDAO

```
package ch.hsluw.mangelmanager.persister.dao.projekt;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Projekt;

/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ProjektDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Projekt entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Projekt update(Projekt entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Projekt entity) throws Exception;

    void deleteProjektById(Integer id) throws Exception;

    /**
     * Liefert die Projekt-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
```

```
     */
    Projekt findProjektById(Integer id);

    /**
     * Liefert alle Projekt-Objekte zurück.
     *
     * @return
     */
    List<Projekt> findAllProjekt();

    /**
     * Liefert die Liste mit Projekten für die übergebene Bezeichnung
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param bezeichnung
     * @return
     */
    public List<Projekt> findProjektByBezeichnung(String bezeichnung);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Projektstatus
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param projektstatus
     * @return
     */
    public List<Projekt> findProjektByProjektstatus(String projektstatus);

    /**
     * Liefert die Liste mit Projekten für dem übergebenen Ort zurück,
falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param ort
     * @return
     */
    public List<Projekt> findProjektByOrt(String ort);

    /**
     * Liefert die Liste mit Projekten für dem übergebenen Plz zurück,
falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param plz
     * @return
     */
    public List<Projekt> findProjektByPlz(String plz);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Bauherren
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param bauherr
     * @return
     */
    public List<Projekt> findProjektByBauherr(String bauherr);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Objekttyp
zurück, falls
```

```
     * welche gefunden, sonst eine leere Liste.
     *
     * @param objekttyp
     * @return
     */
    public List<Projekt> findProjektByObjekttyp(String objekttyp);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Arbeitstyp
zurück, falls
     * welche gefunden, sonst eine leere Liste.
     *
     * @param arbeitstyp
     * @return
     */
    public List<Projekt> findProjektByArbeitstyp(String arbeitstyp);

    /**
     * Liefert die Liste mit Projekten für den übergebenen Zeitrahmen.
     *
     * @param fromDatum
     * @param endDatum
     * @return
     */

    public List<Projekt> findAllSubunternehmenProjekt(Integer
subunternehmen);

    public List<Projekt> findProjektByPerson(Person person);
}
```

ProjektDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.projekt;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface fuer Projekt Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
```

```
public class ProjektDAOImpl implements ProjektDAO {
    @Override
    public void save(Projekt entity) throws Exception {
        new GenericPersisterImpl<Projekt>(Projekt.class).save(entity);
    }

    @Override
    public Projekt update(Projekt entity) throws Exception {
        return new
GenericPersisterImpl<Projekt>(Projekt.class).update(entity);
    }

    @Override
    public void delete(Projekt entity) throws Exception {
        new GenericPersisterImpl<Projekt>(Projekt.class).delete(entity);
    }

    @Override
    public void deleteProjektById(Integer id) throws Exception {
        new GenericPersisterImpl<Projekt>(Projekt.class).deleteById(id);
    }

    @Override
    public Projekt findProjektById(Integer id) {
        return new
GenericPersisterImpl<Projekt>(Projekt.class).findById(id);
    }

    @Override
    public List<Projekt> findAllProjekt() {
        return new GenericPersisterImpl<Projekt>(Projekt.class).findAll();
    }

    @Override
    public List<Projekt> findProjektByBezeichnung(String bezeichnung) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByBezeichnung",
                Projekt.class);

        tQuery.setParameter("bezeichnung", "%" +bezeichnung + "%");

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }

    @Override
    public List<Projekt> findProjektByProjektstatus(String projektstatus) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByProjektstatus",
                Projekt.class);

        tQuery.setParameter("projektstatus", "%" +projektstatus +"%");
```

```
        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }

    @Override
    public List<Projekt> findProjektByOrt(String ort) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByOrt",
                Projekt.class);

        tQuery.setParameter("ort", "%" + ort + "%");

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }

    @Override
    public List<Projekt> findProjektByPlz(String plz) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByPlz",
                Projekt.class);

        tQuery.setParameter("plz", Integer.parseInt(plz));

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }

    @Override
    public List<Projekt> findProjektByBauherr(String bauherr) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByBauherr",
                Projekt.class);

        tQuery.setParameter("bauherr", "%" +bauherr + "%");

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
```

```
    }

    @Override
    public List<Projekt> findProjektByObjekttyp(String objekttyp) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByObjekttyp",
                Projekt.class);

        tQuery.setParameter("objekttyp", "%" + objekttyp + "%");

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }

    @Override
    public List<Projekt> findProjektByArbeitstyp(String arbeitstyp) {

        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<Projekt> tQuery =
em.createNamedQuery("Projekt.findByArbeitstyp",
                Projekt.class);

        tQuery.setParameter("arbeitstyp", "%" +arbeitstyp+ "%");

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }



    @Override
    public List<Projekt> findAllSubunternehmenProjekt(Integer
subunternehmen) {
        EntityManager em = JpaUtil.createEntityManager();

        Query tQuery = em.createNativeQuery("select distinct p.* from
projekt as p, projektsumitarbeiter as ps, "
                + "sumitarbeiter as sm "
                + "where ps.fkprojekt_id = p.id "
                + "and ps.fkmitarbeiter_id = sm.id "
                + "and sm.fksubunternehmen_id = "+subunternehmen,
                Projekt.class);

        List<Projekt> projektListe = tQuery.getResultList();

        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
    }
```

```
    @Override
    public List<Projekt> findProjektByPerson(Person person) {
EntityManager em = JpaUtil.createEntityManager();

        if(person instanceof SuMitarbeiter){
        Query tQuery = em.createNativeQuery("select distinct p.* from
projekt as p,"
                + " projektsumitarbeiter as ps, person as pr "
                + " where ps.fkprojekt_id = p.id "
                + " and ps.fkmitarbeiter_id = pr.id"
                + " and pr.id = "+person.getId(),
                Projekt.class);
        List<Projekt> projektListe = tQuery.getResultList();
        em.close();

        return projektListe != null ? projektListe : new
ArrayList<Projekt>();
        }
        else if(person instanceof GuMitarbeiter){
            Query tQuery = em.createNativeQuery("select distinct p.* from
projekt as p,"
                    + " projektgumitarbeiter as pg, person as pr "
                    + " where pg.fkprojekt_id = p.id "
                    + " and pg.fkmitarbeiter_id = pr.id"
                    + " and pr.id = "+person.getId(),
                    Projekt.class);
            List<Projekt> projektListe = tQuery.getResultList();
            em.close();

            return projektListe != null ? projektListe : new
ArrayList<Projekt>();
            }else{
            Query tQuery = em.createNativeQuery("select distinct p.* from
projekt as p, "
                    + "projekt_bauherr as b, person as pr"
                    + " where b.projekt_id = p.id"
                    + " and b.fkbauherr_id = pr.id"
                    + " and pr.id = "+person.getId(),
                    Projekt.class);
            List<Projekt> projektListe = tQuery.getResultList();
            em.close();
            return projektListe != null ? projektListe : new
ArrayList<Projekt>();
        }

    }
}
```

ProjektGuMitarbeiterDAO

```
package ch.hsluw.mangelmanager.persister.dao.projektgumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
```

```java
/**
 * Interface fuer ProjektGuMitarbeiterGuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface ProjektGuMitarbeiterDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(ProjektGuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    ProjektGuMitarbeiter update(ProjektGuMitarbeiter entity) throws
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(ProjektGuMitarbeiter entity) throws Exception;

    void deleteProjektGuMitarbeiterById( Integer idprojekt, Integer
idguMitarbeiter) throws Exception;

    /**
     * Liefert die ProjektGuMitarbeiter-Entity für die uebergebenen Werte
zurück.
     *
     * @param idprojekt
     * @param idguMitarbeiter
     * @return
     */
    ProjektGuMitarbeiter findProjektGuMitarbeiterById(Integer idprojekt,
Integer idguMitarbeiter);

    /**
     * Liefert alle ProjektGuMitarbeiter-Objekte zurück.
     *
     * @return
     */
    List<ProjektGuMitarbeiter> findAllProjektGuMitarbeiter();

    List<ProjektGuMitarbeiter> findAllBauleiterByProjekt(Projekt projekt);
}
```

ProjektGuMitarbeiterDAOImpl

```java
package ch.hsluw.mangelmanager.persister.dao.projektgumitarbeiter;
```

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface implementation fuer ProjektGuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class ProjektGuMitarbeiterDAOImpl implements ProjektGuMitarbeiterDAO
{
    @Override
    public void save(ProjektGuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<ProjektGuMitarbeiter>(ProjektGuMitarbeiter.class).save
(entity);
    }

    @Override
    public ProjektGuMitarbeiter update(ProjektGuMitarbeiter entity) throws
Exception {
        return new
GenericPersisterImpl<ProjektGuMitarbeiter>(ProjektGuMitarbeiter.class).upda
te(entity);
    }

    @Override
    public void delete(ProjektGuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<ProjektGuMitarbeiter>(ProjektGuMitarbeiter.class).dele
te(entity);
    }

    @Override
    public void deleteProjektGuMitarbeiterById(Integer idProjekt, Integer
idMitarbeiter) throws Exception {
        new
GenericPersisterImpl<ProjektGuMitarbeiter>(ProjektGuMitarbeiter.class).dele
teById(idProjekt, idMitarbeiter);
    }

    @Override
    public ProjektGuMitarbeiter findProjektGuMitarbeiterById(Integer
idProjekt, Integer idMitarbeiter) {
        return new
GenericPersisterImpl<ProjektGuMitarbeiter>(ProjektGuMitarbeiter.class).find
ById(idProjekt, idMitarbeiter);
```

```
    }

    @Override
    public List<ProjektGuMitarbeiter> findAllProjektGuMitarbeiter() {
        return new
GenericPersisterImpl<ProjektGuMitarbeiter>(ProjektGuMitarbeiter.class).find
All();
    }

    @Override
    public List<ProjektGuMitarbeiter> findAllBauleiterByProjekt(Projekt
projekt) {
        // TODO Auto-generated method stub
        EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<ProjektGuMitarbeiter> tQuery =
em.createNamedQuery("ProjektGuMitarbeiter.findAllBauleiterByProjekt",
                ProjektGuMitarbeiter.class);

        tQuery.setParameter("projektId", projekt);

        List<ProjektGuMitarbeiter> bauleiterListe = tQuery.getResultList();

        em.close();

        return bauleiterListe != null ? bauleiterListe : new
ArrayList<ProjektGuMitarbeiter>();
    }
}
```

ProjektstatusDAO

```
package ch.hsluw.mangelmanager.persister.dao.projektstatus;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Projektstatus;

/**
 * Interface fuer Projektstatus Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
public interface ProjektstatusDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Projektstatus entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
```

```
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Projektstatus update(Projektstatus entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Projektstatus entity) throws Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteProjektstatusById(Integer id) throws Exception;

    /**
     * Liefert die Projektstatus-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Projektstatus findProjektstatusById(Integer id);

    /**
     * Liefert alle Projektstatus-Objekte zurück.
     *
     * @return
     */
    List<Projektstatus> findAllProjektstatus();


}
```

ProjektstatusDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.projektstatus;

/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

import java.util.List;

import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface fuer Projektstatus Entity
 *
 * @version 1.0
 * @author sritz
 *
 */
```

```
public class ProjektstatusDAOImpl implements ProjektstatusDAO {
    @Override
    public void save(Projektstatus entity) throws Exception {
        new
GenericPersisterImpl<Projektstatus>(Projektstatus.class).save(entity);
    }

    @Override
    public Projektstatus update(Projektstatus entity) throws Exception {
        return new
GenericPersisterImpl<Projektstatus>(Projektstatus.class).update(entity);
    }

    @Override
    public void delete(Projektstatus entity) throws Exception {
        new
GenericPersisterImpl<Projektstatus>(Projektstatus.class).delete(entity);
    }

    @Override
    public void deleteProjektstatusById(Integer id) throws Exception {
        new
GenericPersisterImpl<Projektstatus>(Projektstatus.class).deleteById(id);
    }

    @Override
    public Projektstatus findProjektstatusById(Integer id) {
        return new
GenericPersisterImpl<Projektstatus>(Projektstatus.class).findById(id);
    }

    @Override
    public List<Projektstatus> findAllProjektstatus() {
        return new
GenericPersisterImpl<Projektstatus>(Projektstatus.class).findAll();
    }

}
```

ProjektSuMitarbeiterDAO

```
package ch.hsluw.mangelmanager.persister.dao.projektsumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;

/**
 * Interface fuer ProjektSuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface ProjektSuMitarbeiterDAO {
    /**
     * Speichert die übergebene Entity.
```

```
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(ProjektSuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    ProjektSuMitarbeiter update(ProjektSuMitarbeiter entity) throws
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(ProjektSuMitarbeiter entity) throws Exception;

    void deleteProjektSuMitarbeiterById( Integer idprojekt, Integer
idguMitarbeiter) throws Exception;

    /**
     * Liefert die ProjektSuMitarbeiter-Entity für die uebergebenen Werte
zurück.
     *
     * @param idprojekt
     * @param idguMitarbeiter
     * @return
     */
    ProjektSuMitarbeiter findProjektSuMitarbeiterById(Integer idprojekt,
Integer idguMitarbeiter);

    /**
     * Liefert alle ProjektSuMitarbeiter-Objekte zurück.
     *
     * @return
     */
    List<ProjektSuMitarbeiter> findAllProjektSuMitarbeiter();
}
```

ProjektSuMitarbeiterDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.projektsumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
```

```
 * Interface implementation fuer ProjektSuMitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class ProjektSuMitarbeiterDAOImpl implements ProjektSuMitarbeiterDAO
{
    @Override
    public void save(ProjektSuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<ProjektSuMitarbeiter>(ProjektSuMitarbeiter.class).save
(entity);
    }

    @Override
    public ProjektSuMitarbeiter update(ProjektSuMitarbeiter entity) throws
Exception {
        return new
GenericPersisterImpl<ProjektSuMitarbeiter>(ProjektSuMitarbeiter.class).upda
te(entity);
    }

    @Override
    public void delete(ProjektSuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<ProjektSuMitarbeiter>(ProjektSuMitarbeiter.class).dele
te(entity);
    }

    @Override
    public void deleteProjektSuMitarbeiterById(Integer idProjekt, Integer
idMitarbeiter) throws Exception {
        new
GenericPersisterImpl<ProjektSuMitarbeiter>(ProjektSuMitarbeiter.class).dele
teById(idProjekt, idMitarbeiter);
    }

    @Override
    public ProjektSuMitarbeiter findProjektSuMitarbeiterById(Integer
idProjekt, Integer idMitarbeiter) {
        return new
GenericPersisterImpl<ProjektSuMitarbeiter>(ProjektSuMitarbeiter.class).find
ById(idProjekt, idMitarbeiter);
    }

    @Override
    public List<ProjektSuMitarbeiter> findAllProjektSuMitarbeiter() {
        return new
GenericPersisterImpl<ProjektSuMitarbeiter>(ProjektSuMitarbeiter.class).find
All();
    }
}
```

RolleDAO

```
package ch.hsluw.mangelmanager.persister.dao.rolle;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
import java.util.List;

import ch.hsluw.mangelmanager.model.Rolle;

/**
 * Interface fuer Rolle Entity
 *
 * @version 1.0
 * @author miten
 *
 */
public interface RolleDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Rolle entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    Rolle update(Rolle entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Rolle entity) throws Exception;

    void deleteRolleById(Integer id) throws Exception;

    /**
     * Liefert die Rolle-Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    Rolle findRolleById(Integer id);

    /**
     * Liefert alle Rolle-Objekte zurück.
     *
     * @return
     */
    List<Rolle> findAllRolle();

}
```

RolleDAOImpl

```java
package ch.hsluw.mangelmanager.persister.dao.rolle;
```

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface fuer Rolle Entity
 *
 * @version 1.0
 * @author miten
 *
 */
public class RolleDAOImpl implements RolleDAO {
    @Override
    public void save(Rolle entity) throws Exception {
        new GenericPersisterImpl<Rolle>(Rolle.class).save(entity);
    }

    @Override
    public Rolle update(Rolle entity) throws Exception {
        return new GenericPersisterImpl<Rolle>(Rolle.class).update(entity);
    }

    @Override
    public void delete(Rolle entity) throws Exception {
        new GenericPersisterImpl<Rolle>(Rolle.class).delete(entity);
    }

    @Override
    public void deleteRolleById(Integer id) throws Exception {
        new GenericPersisterImpl<Rolle>(Rolle.class).deleteById(id);
    }

    @Override
    public Rolle findRolleById(Integer id) {
        return new GenericPersisterImpl<Rolle>(Rolle.class).findById(id);
    }

    @Override
    public List<Rolle> findAllRolle() {
        return new GenericPersisterImpl<Rolle>(Rolle.class).findAll();
    }
}
```

SubunternehmenDAO

```java
package ch.hsluw.mangelmanager.persister.dao.subunternehmen;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;
```

```java
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

/**
 * Interface fuer Subunternehmen Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface SubunternehmenDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(Subunternehmen entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void update(Subunternehmen entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(Subunternehmen entity) throws Exception;

    void deleteSubunternehmenById(Integer id) throws Exception;

    /**
     * Liefert die Subunternehmen-Entity für den übergebenen Id-Wert
zurück.
     *
     * @param id
     * @return
     */
    Subunternehmen findSubunternehmenById(Integer id);

    /**
     * Liefert alle Subunternehmen-Objekte zurück.
     *
     * @return
     */
    List<Subunternehmen> findAllSubunternehmen();


    String findAllProjekte(int subunternehmen);

    List<SuMitarbeiter> findAllSubunternehmenMitarbeiter(Subunternehmen
subunternehmen);

    List<Subunternehmen> findAllSubunternehmenByProjekt(Integer projekt2);
```

```
}
```

SubunternehmenDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.subunternehmen;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.persistence.TypedQuery;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Interface implementation fuer Subunternehmen Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class SubunternehmenDAOImpl implements SubunternehmenDAO {
    @Override
    public void save(Subunternehmen entity) throws Exception {
        new
GenericPersisterImpl<Subunternehmen>(Subunternehmen.class).save(entity);
    }

    @Override
    public void update(Subunternehmen entity) throws Exception {
         new
GenericPersisterImpl<Subunternehmen>(Subunternehmen.class).update(entity);
    }

    @Override
    public void delete(Subunternehmen entity) throws Exception {
        new
GenericPersisterImpl<Subunternehmen>(Subunternehmen.class).delete(entity);
    }
    @Override
    public void deleteSubunternehmenById(Integer id) throws Exception {
        new
GenericPersisterImpl<Subunternehmen>(Subunternehmen.class).deleteById(id);
    }

    @Override
    public Subunternehmen findSubunternehmenById(Integer id) {
        return new
GenericPersisterImpl<Subunternehmen>(Subunternehmen.class).findById(id);
    }

    @Override
    public List<Subunternehmen> findAllSubunternehmen() {
```

```
        return new
GenericPersisterImpl<Subunternehmen>(Subunternehmen.class).findAll();
    }

    @Override
    public String findAllProjekte(int subunternehmen) {
        EntityManager em = JpaUtil.createEntityManager();
        String anzProjekte = "";


        Query tQuery = em.createNativeQuery("select count(distinct
ps.fkprojekt_id) from projektsumitarbeiter as ps "
                + "join sumitarbeiter as s on s.id = ps.fkmitarbeiter_id "
                + "join projekt as p on p.id = ps.fkprojekt_id "
                + "join projektstatus pst on pst.id = p.fkprojektstatus_id
"
                + "where s.fksubunternehmen_id = "+ subunternehmen +" and
pst.bezeichnung != 'abgeschlossen'");

        Object resProjekte = tQuery.getSingleResult();

        em.close();
        anzProjekte = resProjekte.toString();

        return anzProjekte;

    }

    @Override
    public List<SuMitarbeiter>
findAllSubunternehmenMitarbeiter(Subunternehmen subunternehmen) {
EntityManager em = JpaUtil.createEntityManager();

        TypedQuery<SuMitarbeiter> tQuery =
em.createNamedQuery("Subunternehmen.findBySubunternehmenMitarbeiter",
                SuMitarbeiter.class);

        tQuery.setParameter("subunternehmenId", subunternehmen);

        List<SuMitarbeiter> mitarbeiterListe = tQuery.getResultList();

        em.close();

        return mitarbeiterListe != null ? mitarbeiterListe : new
ArrayList<SuMitarbeiter>();
    }

    @Override
    public List<Subunternehmen> findAllSubunternehmenByProjekt(Integer
projekt2) {
        // TODO Auto-generated method stub
        EntityManager em = JpaUtil.createEntityManager();


        Query tQuery = em.createNativeQuery("select distinct s.* from
subunternehmen as s, "
                + "sumitarbeiter as sm, projektsumitarbeiter as ps "
                + "where ps.fkmitarbeiter_id = sm.id "
                + "and sm.fksubunternehmen_id = s.id and ps.fkprojekt_id =
"+ projekt2,
                Subunternehmen.class);

        List<Subunternehmen> subunternehmenListe = tQuery.getResultList();
```

```
        em.close();

        return subunternehmenListe != null ? subunternehmenListe : new
ArrayList<Subunternehmen>();
    }

}
```

SuMitarbeiterDAO

```
package ch.hsluw.mangelmanager.persister.dao.sumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;

/**
 * Interface fuer Subunternehmen Mitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public interface SuMitarbeiterDAO {
    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    void save(SuMitarbeiter entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    SuMitarbeiter update(SuMitarbeiter entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(SuMitarbeiter entity) throws Exception;

    void deleteSuMitarbeiterById(Integer id) throws Exception;

    /**
     * Liefert die SuMitarbeiter-Entity für den übergebenen Id-Wert zurück.
     *
```

```
     * @param id
     * @return
     */
    SuMitarbeiter findSuMitarbeiterById(Integer id);

    /**
     * Liefert alle SuMitarbeiter-Objekte zurück.
     *
     * @return
     */
    List<SuMitarbeiter> findAllSuMitarbeiter();
}
```

SuMitarbeiterDAOImpl

```
package ch.hsluw.mangelmanager.persister.dao.sumitarbeiter;
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */



import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.persister.generic.GenericPersisterImpl;

/**
 * Interface implementation fuer Subunternehmen Mitarbeiter Entity
 *
 * @version 1.0
 * @author lkuendig
 *
 */
public class SuMitarbeiterDAOImpl implements SuMitarbeiterDAO {
    @Override
    public void save(SuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).save(entity);
    }

    @Override
    public SuMitarbeiter update(SuMitarbeiter entity) throws Exception {
        return new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).update(entity);
    }

    @Override
    public void delete(SuMitarbeiter entity) throws Exception {
        new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).delete(entity);
    }

    @Override
    public void deleteSuMitarbeiterById(Integer id) throws Exception {
        new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).deleteById(id);
    }

    @Override
    public SuMitarbeiter findSuMitarbeiterById(Integer id) {
```

```
        return new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).findById(id);
    }

    @Override
    public List<SuMitarbeiter> findAllSuMitarbeiter() {
        return new
GenericPersisterImpl<SuMitarbeiter>(SuMitarbeiter.class).findAll();
    }
}
```

GenericPersister

```
/*
 * ZWECK: Referenzprojekt
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 *
 * Copyright (c) Jordan Sucur - Februar 2015
 */
package ch.hsluw.mangelmanager.persister.generic;

import java.util.List;

/**
 * Interface für CRUD-Basisoperationen.
 *
 * @author jsucur
 * @version 1.0
 *
 * @param <T>
 *
 */
public interface GenericPersister<T> {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void save(T entity) throws Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws Exception
     */
    T update(T entity) throws Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws Exception
     */
    void delete(T entity) throws Exception;

    /**
     * Löscht die Entity für den übergebenen Id-Wert.
     *
```

```
     * @param id
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;
    void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     */
    T findById(Integer id);

    T findById(Integer idProjekt, Integer idMitarbeiter);

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     */
    List<T> findAll();

}
```

GenericPersisterImpl

```
/*
 * ZWECK: Referenzprojekt
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 *
 * Copyright (c) Jordan Sucur - Februar 2015
 */
package ch.hsluw.mangelmanager.persister.generic;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.persistence.TypedQuery;

import org.apache.log4j.Logger;

import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Diese Klasse stellt die CRUD-Methoden zur Verfügung, die in allen
 * Unterklassen vorkommen müssen.
 *
 * @version 1.0
 * @author jsucur
 *
 * @param <T>
 *
 */
public class GenericPersisterImpl<T> implements GenericPersister<T> {

    private static final Logger logger = Logger
            .getLogger(GenericPersisterImpl.class);

    protected Class<T> classType;
```

```
    public GenericPersisterImpl(Class<T> type) {
        this.classType = type;
    }

    @Override
    public void save(T entity) throws Exception {

        EntityManager em = JpaUtil.createEntityManager();

        try {
            em.getTransaction().begin();
            em.persist(entity);
            em.getTransaction().commit();
        } catch (Exception e) {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            logger.error("Fehler beim Speichern der Entity vom Typ \'"
                    + classType.getName() + "\': [" + entity.toString() +
"]",
                    e);
            throw e;

        } finally {
            em.close();
        }
    }

    @Override
    public T update(T entity) throws Exception {

        EntityManager em = JpaUtil.createEntityManager();
        em.getTransaction().begin();

        T eMerged = null;

        try {
            eMerged = em.merge(entity);

            logger.info("Entity vom Typ " + classType.getSimpleName()
                    + " wird updatet: " + entity);

            em.getTransaction().commit();
        } catch (Exception e) {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            logger.error(
                    "Fehler beim Update der Entity vom Typ \'"
                            + classType.getName() + "\': [" + entity + "]",
e);
            throw e;

        } finally {
            em.close();
        }

        return eMerged;
    }

    @Override
    public void delete(T entity) throws Exception {
```

```
        EntityManager em = JpaUtil.createEntityManager();
        em.getTransaction().begin();

        try {

            if (em.contains(entity)) {
                em.remove(entity);
            } else {
                em.remove(em.merge(entity));
            }

            logger.info("Entity vom Typ " + classType.getSimpleName() + "
["
                    + entity + "] " + " wird gelöscht");

            em.getTransaction().commit();
        } catch (Exception e) {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }

            logger.error("Fehler beim Löschen der Entity vom Typ \'"
                    + classType.getName() + "\': [" + entity + "]", e);

            throw e;

        } finally {
            em.close();
        }
    }

    @Override
    public void deleteById(Integer id) throws Exception {

        EntityManager em = JpaUtil.createEntityManager();
        String strQuery = "DELETE FROM " + classType.getSimpleName()
                + " entity WHERE entity.id = :id";

        Query query = em.createQuery(strQuery);
        query.setParameter("id", id);

        try {
            em.getTransaction().begin();

            query.executeUpdate();

            logger.info("Entity vom Typ " + classType.getSimpleName()
                    + " mit id = " + id + " wird gelöscht");

            em.getTransaction().commit();
        } catch (Exception e) {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            logger.error("Fehler beim Löschen der Entity vom Typ \'"
                    + classType.getName() + "\': [ id = " + id + "]", e);
            throw e;

        } finally {
            em.close();
        }
    }
```

```
    @Override
    public void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception {

        EntityManager em = JpaUtil.createEntityManager();
        String strQuery = "DELETE FROM " + classType.getSimpleName()
                + " entity WHERE entity.fkprojekt = :idProjekt AND
entity.fkMitarbeiter = :idMitarbeiter";

        Query query = em.createQuery(strQuery);
        query.setParameter("idProjket", idProjekt);
        query.setParameter("idMitarbeiter", idMitarbeiter);

        try {
            em.getTransaction().begin();

            query.executeUpdate();

            logger.info("Entity vom Typ " + classType.getSimpleName()
                    + " mit Projekt id = " + idProjekt + " und Mitarbeiter
id = "+ idMitarbeiter + " wird gelöscht");

            em.getTransaction().commit();
        } catch (Exception e) {
            if (em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            logger.error("Fehler beim Löschen der Entity vom Typ \'"
                    + classType.getName() + "\': [ Projekt id = " +
idProjekt + " und Mitarbeiter id = "+ idMitarbeiter + "]", e);
            throw e;

        } finally {
            em.close();
        }
    }

    @Override
    public T findById(Integer idProjekt, Integer idMitarbeiter) {

        EntityManager em = JpaUtil.createEntityManager();
        String strQuery = "SELECT FROM " + classType.getSimpleName()
                + " entity WHERE entity.fkprojekt = :idProjekt AND
entity.fkMitarbeiter = :idMitarbeiter";

        Query query = em.createQuery(strQuery);
        query.setParameter("idProjket", idProjekt);
        query.setParameter("idMitarbeiter", idMitarbeiter);

        TypedQuery<T> q =
JpaUtil.createEntityManager().createQuery(strQuery, classType);
        return q.getSingleResult();
    }

    @Override
    public T findById(Integer id) {
        return JpaUtil.createEntityManager().find(classType, id);
    }

    @Override
    public List<T> findAll() {
```

```
        String sql = "SELECT entity FROM " + classType.getSimpleName()
                + " entity";
        TypedQuery<T> q = JpaUtil.createEntityManager().createQuery(sql,
                classType);
        return q.getResultList();
    }
}
```

JpaUtil

```
/*
 * ZWECK: Referenzprojekt
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 *
 * Copyright (c) Jordan Sucur - Februar 2015
 */
package ch.hsluw.mangelmanager.persister.util;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 * Diese Klasse bildetet eine Hilfsklasse ab, die sich um die Erstellung
der
 * EntityManager-Instanz kümmert.
 *
 * @version 1.0
 * @author jsucur
 *
 */
public class JpaUtil {

    private static EntityManagerFactory entityManagerFactory = null;

    static {
        try {
            /* EntityManagerFactory erzeugen */
            entityManagerFactory = Persistence
                    .createEntityManagerFactory("MangelManagerPU");
        } catch (Throwable e) {
            /* TODO - Fehlerbehandlung … */
            e.printStackTrace();
        }
    }

    public static EntityManager createEntityManager() {
        return entityManagerFactory.createEntityManager();
    }
}
```

CreateEntityTest

```
package ch.hsluw.mangelmanager.persister;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;
```

```java
import javax.persistence.EntityManager;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;
import org.junit.Test;

import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
import ch.hsluw.mangelmanager.persister.util.JpaUtil;

/**
 * Diese Klasse füllt die Datenbank mit Daten.
 *
 * @version 1.0
 * @author mmont
 *
 */

public class CreateEntityTest {


    /**
     * Create empty Lists
     */
    List<Adresse> listAdresse = null;
    List<Arbeitstyp> listArbeitstyp = null;
    List<Bauherr> listBauherr = null;
    List<Bauherr> listBauherr2 = null;
    List<Bauherr> listBauherr3 = null;
    List<Bauherr> listBauherr4 = null;
    List<Bauherr> listBauherr5 = null;
    List<GuMitarbeiter> listGuMitarbeiter = null;
    List<Login> listLogin = null;
    List<Mangel> listMangel = null;
    List<Mangelstatus> listMangelstatus = null;
    List<Meldung> listMeldung = null;
    List<Meldungstyp> listMeldungstyp = null;
    List<Objekttyp> listObjekttyp = null;
    List<Person> listPerson = null;
    List<Plz> listPlz = null;
    List<Projekt> listProjekt = null;
    List<Projektstatus> listProjektstatus = null;
    List<ProjektGuMitarbeiter> listProjektGuMitarbeiter = null;
    List<ProjektSuMitarbeiter> listProjektSuMitarbeiter = null;
    List<Rolle> listRolle = null;
```

```
    List<Subunternehmen> listSubunternehmen = null;
    List<SuMitarbeiter> listSuMitarbeiter = null;
    EntityManager em = null;

    /**
     * Change lists to ArrayLists and fill them with Data
     */
    @Test
    public void fillLists() {
        listAdresse = new ArrayList<Adresse>();
        listArbeitstyp = new ArrayList<Arbeitstyp>();
        listBauherr = new ArrayList<Bauherr>();
        listBauherr2 = new ArrayList<Bauherr>();
        listBauherr3 = new ArrayList<Bauherr>();
        listBauherr4 = new ArrayList<Bauherr>();
        listBauherr5 = new ArrayList<Bauherr>();
        listGuMitarbeiter = new ArrayList<GuMitarbeiter>();
        listLogin = new ArrayList<Login>();
        listMangel = new ArrayList<Mangel>();
        listMangelstatus = new ArrayList<Mangelstatus>();
        listMeldung = new ArrayList<Meldung>();
        listMeldungstyp = new ArrayList<Meldungstyp>();
        listObjekttyp = new ArrayList<Objekttyp>();
        listPerson = new ArrayList<Person>();
        listPlz = new ArrayList<Plz>();
        listProjekt = new ArrayList<Projekt>();
        listProjektstatus = new ArrayList<Projektstatus>();
        listProjektGuMitarbeiter = new ArrayList<ProjektGuMitarbeiter>();
        listProjektSuMitarbeiter = new ArrayList<ProjektSuMitarbeiter>();
        listRolle = new ArrayList<Rolle>();
        listSubunternehmen = new ArrayList<Subunternehmen>();
        listSuMitarbeiter = new ArrayList<SuMitarbeiter>();

        /**
         * fill listPlz from csv Ortschaften.csv with PLZ und
Ortschaftsnamen
         */
        try {
            FileReader fileread = new FileReader("Ortschaften.csv");
            CSVParser parser = new CSVParser(fileread, CSVFormat.EXCEL
                    .withDelimiter(';').withHeader());
            for (CSVRecord r : parser) {
                listPlz.add(new Plz(Integer.parseInt(r.get("PLZ")), r
                        .get("Ortschaftsname")));

            }
            parser.close();

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        /**
         * Fill Lists with data
         */
        listAdresse.add(new Adresse("Baustrasse 1", listPlz.get(2)));
        listAdresse.add(new Adresse("Baustrasse 2", listPlz.get(2)));
        listAdresse.add(new Adresse("Baustrasse 3", listPlz.get(2)));
        listAdresse.add(new Adresse("Baustrasse 4", listPlz.get(2)));
        listAdresse.add(new Adresse("Kirchstrasse 2", listPlz.get(5)));
        listAdresse.add(new Adresse("Riedtli 3", listPlz.get(4)));
```

```
        listAdresse.add(new Adresse("Aarauigen 2", listPlz.get(4)));
        listAdresse.add(new Adresse("Hauptstrasse 123", listPlz.get(4)));
        listAdresse.add(new Adresse("Dorfbach 49", listPlz.get(10)));
        listAdresse.add(new Adresse("Waldstrasse 2", listPlz.get(15)));
        listAdresse.add(new Adresse("Kupferbach 3", listPlz.get(250)));
        listAdresse.add(new Adresse("Kupferbach 5", listPlz.get(250)));
        listAdresse.add(new Adresse("Kupferbach 23", listPlz.get(250)));
        listAdresse.add(new Adresse("Weitenberg 20", listPlz.get(340)));
        listAdresse.add(new Adresse("Weitenberg 50", listPlz.get(340)));
        listAdresse.add(new Adresse("Malberg 4", listPlz.get(890)));
        listAdresse.add(new Adresse("Biereberg 4", listPlz.get(432)));
        listAdresse.add(new Adresse("Bergberg 5", listPlz.get(523)));
        listAdresse.add(new Adresse("Seelisberg 3", listPlz.get(514)));
        listAdresse.add(new Adresse("Steinenberg 2", listPlz.get(522)));
        listAdresse.add(new Adresse("Kevin Stadelmannstrasse 40", listPlz
                .get(1)));
        listAdresse
                .add(new Adresse("Stefan Beelerstrasse 23",
listPlz.get(25)));

        listArbeitstyp.add(new Arbeitstyp("Umbau"));
        listArbeitstyp.add(new Arbeitstyp("Neubau"));
        listArbeitstyp.add(new Arbeitstyp("Renovation"));
        listArbeitstyp.add(new Arbeitstyp("Teil-Renovation"));

        listBauherr.add(new Bauherr("Josef", "Schmid", "0774440303",
                listAdresse.get(0)));
        listBauherr.add(new Bauherr("Cihan", "Demir", "0779003003",
listAdresse
                .get(1)));
        listBauherr2.add(new Bauherr("Mike", "Iten", "0764903838",
listAdresse
                .get(2)));
        listBauherr3.add(new Bauherr("Jonas", "Justus", "0902003039",
                listAdresse.get(3)));
        listBauherr3.add(new Bauherr("Bernanrd", "Berg", "0393004039",
                listAdresse.get(4)));
        listBauherr4.add(new Bauherr("Franz", "Tomini", "0498499090",
                listAdresse.get(5)));
        listBauherr5.add(new Bauherr("Kurt", "Kugler", "0498495050",
                listAdresse.get(6)));
        listBauherr5.add(new Bauherr("Kurt", "Kert", "0484891350",
listAdresse
                .get(7)));

        listProjektstatus.add(new Projektstatus("Offen"));
        listProjektstatus.add(new Projektstatus("Abgeschlossen"));

        listRolle.add(new Rolle("Sachbearbeiter"));
        listRolle.add(new Rolle("Bauleiter"));
        listRolle.add(new Rolle("Ansprechsperson"));
        listRolle.add(new Rolle("Subunternehmen-Sachbearbeiter"));

        listLogin.add(new Login("sachbearbeiter", "sachbearbeiter",
                "sachbearbeiter@mangel.ch", listRolle.get(0)));
        listLogin.add(new Login("bauleiter", "bauleiter",
                "bauleiter@mangel.ch", listRolle.get(1)));
        listLogin.add(new Login("ansprechsperson", "ansprechsperson",
                "ansprechsperson@mangel.ch", listRolle.get(2)));
        listLogin.add(new Login("subunternehmen", "subunternehmen",
                "subunternehmen@mangel.ch", listRolle.get(3)));
        listLogin.add(new Login("gumitarbeiter1", "gumitarbeiter1",
                "mitarbeiter@mangel.ch", listRolle.get(1)));
```

```
listLogin.add(new Login("gumitarbeiter2", "gumitarbeiter2",
        "mitarbeiter2@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter3", "gumitarbeiter3",
        "mitarbeiter3@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter4", "gumitarbeiter4",
        "mitarbeiter4@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter5", "gumitarbeiter5",
        "mitarbeiter5@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter6", "gumitarbeiter6",
        "mitarbeiter6@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter7", "gumitarbeiter7",
        "mitarbeiter7@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter8", "gumitarbeiter8",
        "mitarbeiter8@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter9", "gumitarbeiter9",
        "mitarbeiter9@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter10", "gumitarbeiter10",
        "mitarbeiter10@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("gumitarbeiter11", "gumitarbeiter11",
        "mitarbeiter11@mangel.ch", listRolle.get(1)));
listLogin.add(new Login("Gellhorn", "sumitarbeiter",
        "sumitarbeiter@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter1", "sumitarbeiter1",
        "sumitarbeiter1@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter2", "sumitarbeiter2",
        "sumitarbeiter2@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter3", "sumitarbeiter3",
        "sumitarbeiter3@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter4", "sumitarbeiter4",
        "sumitarbeiter4@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter5", "sumitarbeiter5",
        "sumitarbeiter5@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter6", "sumitarbeiter6",
        "sumitarbeiter6@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter7", "sumitarbeiter7",
        "sumitarbeiter7@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter8", "sumitarbeiter8",
        "sumitarbeiter98@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter9", "sumitarbeiter9",
        "sumitarbeiter10@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter10", "sumitarbeiter10",
        "sumitarbeiter11@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter11", "sumitarbeiter11",
        "sumitarbeiter12@mangel.ch", listRolle.get(3)));
listLogin.add(new Login("sumitarbeiter12", "sumitarbeiter12",
        "sumitarbeiter13@mangel.ch", listRolle.get(3)));

listMangelstatus.add(new Mangelstatus("Offen"));
listMangelstatus.add(new Mangelstatus("Abgeschlossen"));

listGuMitarbeiter.add(new GuMitarbeiter("von Rotz", "Horst",
        "0493904949", listLogin.get(4)));
listGuMitarbeiter.add(new GuMitarbeiter("Baum", "Bauberg",
        "0493904529", listLogin.get(5)));
listGuMitarbeiter.add(new GuMitarbeiter("Berdior", "Megamann",
        "9043034434", listLogin.get(6)));
listGuMitarbeiter.add(new GuMitarbeiter("Bieber", "Horst",
        "0418393939", listLogin.get(7)));
listGuMitarbeiter.add(new GuMitarbeiter("Hummel", "Kuchen",
        "8934932304", listLogin.get(8)));
listGuMitarbeiter.add(new GuMitarbeiter("Rotzer", "Josef",
        "0942384938", listLogin.get(9)));
listGuMitarbeiter.add(new GuMitarbeiter("Rotzberg", "Josefina",
```

```
                "0942384939", listLogin.get(10)));
        listGuMitarbeiter.add(new GuMitarbeiter("von Rotz", "Maria",
                "0942384935", listLogin.get(11)));
        listGuMitarbeiter.add(new GuMitarbeiter("Stadelmann", "Eva",
                "0942384934", listLogin.get(12)));
        listGuMitarbeiter.add(new GuMitarbeiter("Stadelmann", "Angelina",
                "0942384932", listLogin.get(13)));
        listGuMitarbeiter.add(new GuMitarbeiter("Stadelmann", "Kevin",
                "0942384931", listLogin.get(14)));

        listMeldungstyp.add(new Meldungstyp("Information"));
        listMeldungstyp.add(new Meldungstyp("Reklamation"));

        listObjekttyp.add(new Objekttyp("Einfamilienhaus"));
        listObjekttyp.add(new Objekttyp("Mehrfamilienhaus"));
        listObjekttyp.add(new Objekttyp("Garage"));
        listObjekttyp.add(new Objekttyp("Wohnung"));
        listObjekttyp.add(new Objekttyp("Hütte"));
        listObjekttyp.add(new Objekttyp("Untergrund"));
        listObjekttyp.add(new Objekttyp("Industriegebäude"));

        listProjekt.add(new Projekt(listAdresse.get(8), "Projekt Alpha",
                listBauherr, new GregorianCalendar(2015, 4, 14),
                new GregorianCalendar(2015, 6, 06), listObjekttyp.get(0),
                listArbeitstyp.get(1), new GregorianCalendar(2015, 6, 06),
                listProjektstatus.get(1)));
        listProjekt.add(new Projekt(listAdresse.get(8), "Projekt Beta",
                listBauherr2, new GregorianCalendar(2015, 4, 01), null,
                listObjekttyp.get(2), listArbeitstyp.get(0),
                new GregorianCalendar(2015, 7, 06),
listProjektstatus.get(0)));
        listProjekt.add(new Projekt(listAdresse.get(10), "Projekt Gamma",
                listBauherr3, new GregorianCalendar(2015, 4, 30), null,
                listObjekttyp.get(3), listArbeitstyp.get(1),
                new GregorianCalendar(2015, 7, 15),
listProjektstatus.get(0)));
        listProjekt.add(new Projekt(listAdresse.get(11), "Projekt Delta",
                listBauherr4, new GregorianCalendar(2015, 1, 01), null,
                listObjekttyp.get(4), listArbeitstyp.get(3),
                new GregorianCalendar(2015, 7, 25),
listProjektstatus.get(0)));
        listProjekt.add(new Projekt(listAdresse.get(12), "Proejkt OMEGA",
                listBauherr5, new GregorianCalendar(2014, 4, 01), null,
                listObjekttyp.get(5), listArbeitstyp.get(0),
                new GregorianCalendar(2015, 7, 20),
listProjektstatus.get(0)));
        listProjekt.add(new Projekt(listAdresse.get(13), "Projekt 9/11",
                listBauherr, new GregorianCalendar(2015, 9, 11), null,
                listObjekttyp.get(2), listArbeitstyp.get(2),
                new GregorianCalendar(2015, 3, 30),
listProjektstatus.get(0)));

        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(0), listGuMitarbeiter.get(0), new
GregorianCalendar(2015,
                4, 14), new GregorianCalendar(2015, 6, 06)));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(0), listGuMitarbeiter.get(1), new
GregorianCalendar(2015,
                4, 14), new GregorianCalendar(2015, 6, 06)));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(0), listGuMitarbeiter.get(2), new
GregorianCalendar(2015,
```

```
                4, 14), new GregorianCalendar(2015, 6, 06)));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(0), listGuMitarbeiter.get(8), new
GregorianCalendar(2015,
                4, 14), new GregorianCalendar(2015, 6, 06)));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(0), listGuMitarbeiter.get(9), new
GregorianCalendar(2015,
                4, 14), new GregorianCalendar(2015, 6, 06)));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(1), listGuMitarbeiter.get(3), new
GregorianCalendar(2015,
                7, 06), null));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(2), listGuMitarbeiter.get(4), new
GregorianCalendar(2015,
                7, 15), null));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(3), listGuMitarbeiter.get(5), new
GregorianCalendar(2015,
                7, 25), null));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(4), listGuMitarbeiter.get(6), new
GregorianCalendar(2015,
                7, 20), null));
        listProjektGuMitarbeiter.add(new ProjektGuMitarbeiter(listProjekt
                .get(5), listGuMitarbeiter.get(7), new
GregorianCalendar(2015,
                3, 30), null));

        listSubunternehmen.add(new Subunternehmen(listAdresse.get(14),
                "Janik von Rotz GmbH", "09006665544"));
        listSubunternehmen.add(new Subunternehmen(listAdresse.get(15),
                "Bosch AG", "09006663344"));
        listSubunternehmen.add(new Subunternehmen(listAdresse.get(16),
                "Mangel AG", "09006662244"));
        listSubunternehmen.add(new Subunternehmen(listAdresse.get(17),
                "Unternehmen AG", "09006661144"));
        listSubunternehmen.add(new Subunternehmen(listAdresse.get(18),
                "Ultra Mangel AG", "09005565544"));
        listSubunternehmen.add(new Subunternehmen(listAdresse.get(19),
                "Destructoid GmbH", "09003465544"));
        listSubunternehmen.add(new Subunternehmen(listAdresse.get(20),
                "Zod AG", "09002265544"));

        listSuMitarbeiter.add(new SuMitarbeiter("von Gellhorn", "Max",
                "09003042030", listSubunternehmen.get(0),
listLogin.get(16)));
        listSuMitarbeiter.add(new SuMitarbeiter("Rotzi", "Janik",
"0418304234",
                listSubunternehmen.get(0), listLogin.get(17)));
        listSuMitarbeiter.add(new SuMitarbeiter("Kündig", "Michael",
                "0418305555", listSubunternehmen.get(5),
listLogin.get(18)));
        listSuMitarbeiter.add(new SuMitarbeiter("Ritz", "Notter",
"0418334234",
                listSubunternehmen.get(4), listLogin.get(19)));
        listSuMitarbeiter.add(new SuMitarbeiter("Bekcic", "Momo",
"0418305234",
                listSubunternehmen.get(0), listLogin.get(20)));
        listSuMitarbeiter.add(new SuMitarbeiter("Beeler", "Stefan",
                "09003042030", listSubunternehmen.get(1),
listLogin.get(21)));
```

```
        listSuMitarbeiter.add(new SuMitarbeiter("Beeler", "Stefan",
                "09003042030", listSubunternehmen.get(1),
listLogin.get(22)));
        listSuMitarbeiter.add(new SuMitarbeiter("Beeler", "Stefan",
                "09003042030", listSubunternehmen.get(2),
listLogin.get(23)));
        listSuMitarbeiter.add(new SuMitarbeiter("Beeler", "Stefan",
                "09003042030", listSubunternehmen.get(3),
listLogin.get(24)));
        listSuMitarbeiter.add(new SuMitarbeiter("Beeler", "Stefan",
                "09003042030", listSubunternehmen.get(0),
listLogin.get(25)));
        listSuMitarbeiter.add(new SuMitarbeiter("Berger", "Stefan",
                "09003042030", listSubunternehmen.get(6),
listLogin.get(15)));

        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(0), listSuMitarbeiter.get(0), new
GregorianCalendar(2015,
                4, 14), new GregorianCalendar(2015, 5, 06)));
        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(0), listSuMitarbeiter.get(1), new
GregorianCalendar(2015,
                4, 14), null));
        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(1), listSuMitarbeiter.get(2), new
GregorianCalendar(2015,
                4, 01), null));
        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(2), listSuMitarbeiter.get(3), new
GregorianCalendar(2015,
                4, 30), null));
        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(3), listSuMitarbeiter.get(4), new
GregorianCalendar(2015,
                1, 01), null));
        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(4), listSuMitarbeiter.get(5), new
GregorianCalendar(2014,
                4, 01), null));
        listProjektSuMitarbeiter.add(new ProjektSuMitarbeiter(listProjekt
                .get(5), listSuMitarbeiter.get(6), new
GregorianCalendar(2015,
                9, 11), null));

        listMangel.add(new Mangel(listProjekt.get(0), "Alpha",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(15),
                "Dachrinne verstopft"));
        listMangel.add(new Mangel(listProjekt.get(0), "Beta",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(15),
                "Dachrinne abgefallen"));
        listMangel.add(new Mangel(listProjekt.get(0), "Rotz",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(15),
                "Wand eingestürtzt"));
```

```
        listMangel.add(new Mangel(listProjekt.get(0), "Beeler",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(1),
listLogin.get(15),
                "Staubsauger verstopft"));
        listMangel.add(new Mangel(listProjekt.get(0), "Bekcic",
                new GregorianCalendar(2015, 5, 05),

                new GregorianCalendar(2015, 06, 01),
listMangelstatus.get(1),
                listLogin.get(15), "Ventilator kaputt"));
        listMangel.add(new Mangel(listProjekt.get(1), "Lothar",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(16),
                "Hier könnte ihre Werbung stehen"));
        listMangel.add(new Mangel(listProjekt.get(2), "Demogramma",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(17),
                "Hier könnte ihre Werbung stehen"));
        listMangel.add(new Mangel(listProjekt.get(3), "Demogrummu",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(18),
                "Hier könnte ihre Werbung stehen"));
        listMangel.add(new Mangel(listProjekt.get(4), "Projektproblem1",
                new GregorianCalendar(2015, 5, 05), new
GregorianCalendar(2015,
                        06, 01), listMangelstatus.get(0),
listLogin.get(19),
                "Hier könnte ihre Werbung stehen"));

        listMeldung.add(new Meldung(listMangel.get(0),
listMeldungstyp.get(1),
                "Bitte schnell beheben", new GregorianCalendar(2015, 5,
05),
                false, listLogin.get(15)));
        listMeldung.add(new Meldung(listMangel.get(1),
listMeldungstyp.get(0),
                "Besorge eine neue Dachrinne", new GregorianCalendar(2015,
5,
                        05), false, listLogin.get(15)));
        listMeldung.add(new Meldung(listMangel.get(2),
listMeldungstyp.get(1),
                "Es zieht", new GregorianCalendar(2015, 5, 05), false,
                listLogin.get(15)));
        listMeldung.add(new Meldung(listMangel.get(3),
listMeldungstyp.get(0),
                "Krieg ihn nicht mehr raus",
                new GregorianCalendar(2015, 5, 05), true,
listLogin.get(15)));
        listMeldung.add(new Meldung(listMangel.get(4),
listMeldungstyp.get(1),
                "Es ist zu heiss!!!", new GregorianCalendar(2015, 5, 05),
true,
                listLogin.get(15)));
        /**
         * create Entity Manger
```

```
 */
em = JpaUtil.createEntityManager();

/**
 * Persists the lists
 */
em.getTransaction().begin();
for (Rolle rolle : listRolle) {
    em.persist(rolle);
}
for (Login login : listLogin) {
    em.persist(login);
}
for (Mangelstatus mangelstatus : listMangelstatus) {
    em.persist(mangelstatus);
}
for (Meldungstyp meldungstyp : listMeldungstyp) {
    em.persist(meldungstyp);
}
for (Plz plz : listPlz) {
    em.persist(plz);
}
for (Adresse adresse : listAdresse) {
    em.persist(adresse);
}
for (Arbeitstyp arbeitstyp : listArbeitstyp) {
    em.persist(arbeitstyp);
}
for (Objekttyp objekttyp : listObjekttyp) {
    em.persist(objekttyp);
}
for (Projektstatus projektstatus : listProjektstatus) {
    em.persist(projektstatus);
}
for (Bauherr bauherr : listBauherr) {
    em.persist(bauherr);
}
for (Bauherr bauherr : listBauherr2) {
    em.persist(bauherr);
}
for (Bauherr bauherr : listBauherr3) {
    em.persist(bauherr);
}
for (Bauherr bauherr : listBauherr4) {
    em.persist(bauherr);
}
for (Bauherr bauherr : listBauherr5) {
    em.persist(bauherr);
}
for (Subunternehmen subunternehmen : listSubunternehmen) {
    em.persist(subunternehmen);
}
for (SuMitarbeiter sumitarbeiter : listSuMitarbeiter) {
    em.persist(sumitarbeiter);
}
for (GuMitarbeiter guMitarbeiter : listGuMitarbeiter) {
    em.persist(guMitarbeiter);
}
for (Person person : listPerson) {
    em.persist(person);
}
for (Projekt projekt : listProjekt) {
    em.persist(projekt);
```

```
        }
        for (ProjektGuMitarbeiter projektGuMitarbeiter :
listProjektGuMitarbeiter) {
            em.persist(projektGuMitarbeiter);
        }

        for (ProjektSuMitarbeiter projektSuMitarbeiter :
listProjektSuMitarbeiter) {
            em.persist(projektSuMitarbeiter);
        }

        for (Mangel mangel : listMangel) {
            em.persist(mangel);
        }

        for (Meldung meldung : listMeldung) {
            em.persist(meldung);
        }

        em.getTransaction().commit();
    }
}
```

EntityTest

```
package ch.hsluw.mangelmanager.persister;

import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.fail;

import java.util.GregorianCalendar;
import java.util.List;

import javax.persistence.RollbackException;

import org.eclipse.persistence.exceptions.DatabaseException;
import org.junit.Test;

import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.persister.dao.adresse.AdresseDAO;
import ch.hsluw.mangelmanager.persister.dao.adresse.AdresseDAOImpl;
import ch.hsluw.mangelmanager.persister.dao.arbeitstyp.ArbeitstypDAO;
import ch.hsluw.mangelmanager.persister.dao.arbeitstyp.ArbeitstypDAOImpl;
import ch.hsluw.mangelmanager.persister.dao.bauherr.BauherrDAO;
import ch.hsluw.mangelmanager.persister.dao.bauherr.BauherrDAOImpl;
import ch.hsluw.mangelmanager.persister.dao.objekttyp.ObjekttypDAO;
import ch.hsluw.mangelmanager.persister.dao.objekttyp.ObjekttypDAOImpl;
import ch.hsluw.mangelmanager.persister.dao.plz.PlzDAO;
import ch.hsluw.mangelmanager.persister.dao.plz.PlzDAOImpl;
import ch.hsluw.mangelmanager.persister.dao.projekt.ProjektDAO;
import ch.hsluw.mangelmanager.persister.dao.projekt.ProjektDAOImpl;
import ch.hsluw.mangelmanager.persister.dao.projektstatus.ProjektstatusDAO;
import
ch.hsluw.mangelmanager.persister.dao.projektstatus.ProjektstatusDAOImpl;
```

```java
/**
 * Diese Klasse testet die Methoden von ProjektDAO und ProjektDAOImpl
 *
 * @version 1.0
 * @author mmont
 *
 */

public class EntityTest {
    /**
     * Tests if Adressen are updateable
     */
    @Test
    public void adressenUpdateTest() {
        try {
            AdresseDAO adressen = new AdresseDAOImpl();
            List<Adresse> listAdresse = adressen.findAllAdresse();
            Adresse adresse = listAdresse.get(listAdresse.size() - 1);
            adresse.setStrasse("testStrasse");
            adressen.update(adresse);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update AdressStrasse");
        }
    }
    /**
     * Tests if Projekte are updateable
     */
    @Test
    public void projektUpdateTest() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            List<Projekt> listProjekt = projektdao.findAllProjekt();
            Projekt projekt = listProjekt.get(listProjekt.size() - 1);
            projekt.setBezeichnung("TestName");
            projektdao.update(projekt);

        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't update ProjektBezeichnung");
        }
    }
    /**
     * Tests if Projekte & Adressen are addable
     */
    @Test
    public void projektAdresseSaveTest() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            PlzDAO plzdao = new PlzDAOImpl();
            List<Plz> listPlz = plzdao.findAllPlz();

            BauherrDAO bauherrdao = new BauherrDAOImpl();
            List<Bauherr> listBauherr = bauherrdao.findAllBauherr();

            ProjektstatusDAO projektstatusdao = new ProjektstatusDAOImpl();
            List<Projektstatus> listProjektstatus = projektstatusdao
                    .findAllProjektstatus();

            ObjekttypDAO objekttypdao = new ObjekttypDAOImpl();
            List<Objekttyp> listObjekttyp =
objekttypdao.findAllObjekttyp();
```

```
            ArbeitstypDAO arbeitstypdao = new ArbeitstypDAOImpl();
            List<Arbeitstyp> listArbeitstyp =
arbeitstypdao.findAllArbeitstyp();

            AdresseDAO adressdao = new AdresseDAOImpl();
            Adresse adresse = new Adresse("Test Strasse 12",
listPlz.get(99));
            adressdao.save(adresse);

            Projekt projekt = new Projekt(adresse, "Projekt Test",
listBauherr,
                    new GregorianCalendar(2014, 4, 01), null,
                    listObjekttyp.get(5), listArbeitstyp.get(0),
                    new GregorianCalendar(2015, 4, 20),
                    listProjektstatus.get(0));

            projektdao.save(projekt);

        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't save Projekt / Adresse");
        }
    }

    /**
     * Test if Projekte are delete able and expects an RollbackException
     *
     * @throws Exception
     */

    @Test(expected = RollbackException.class)
    public void deleteProjekt() throws Exception {
        ProjektDAO projektdao = new ProjektDAOImpl();
        List<Projekt> listProjekt = projektdao.findAllProjekt();
        Projekt projekt = listProjekt.get(listProjekt.size() - 1);
        projektdao.delete(projekt);
    }

    /**
     * Test if Projekte are deleteable by ID and throws a DtabaseException
     *
     * @throws Exception
     */
    @Test(expected = DatabaseException.class)
    public void deleteProjektById() throws Exception {
        ProjektDAO projektdao = new ProjektDAOImpl();
        projektdao.deleteProjektById(109);

    }

    /**
     * Test if Projekte are findable by ID
     */
    @Test
    public void findProjektById() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();

assertNotNull(projektdao.findProjektById(109).getBezeichnung());
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by ID");
        }
```

```
    }
    /**
     * Tests if Projekte are findable by Bezeichnung
     */
    @Test
    public void findProjektByBezeichnung() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();

assertTrue(projektdao.findProjektByBezeichnung(projektdao.findAllProjekt().
get(0).getBezeichnung())
                    .size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Bezeichnung");
        }
    }
    /**
     * Tests if Projekte are findable by Projektstatus
     */
    @Test
    public void findProjektByProjektstatus() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();

assertTrue(projektdao.findProjektByProjektstatus("Abgeschlossen")
                    .size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Projektstatus");
        }
    }
    /**
     * Tests if Projekte are findable by Ort
     */
    @Test
    public void findProjektByOrt() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            List<Projekt> listProjekt = projektdao.findAllProjekt();
            Projekt projekt = listProjekt.get(listProjekt.size() - 1);
            assertTrue(projektdao.findProjektByOrt(
                    projekt.getFkAdresse().getPlz().getOrt()).size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Ort");
        }
    }
    /**
     * Tests if Projekte are findable by Plz
     */
    @Test
    public void findProjektByPlz() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            List<Projekt> listProjekt = projektdao.findAllProjekt();
            Projekt projekt = listProjekt.get(listProjekt.size() - 1);
            assertTrue(projektdao.findProjektByPlz(

projekt.getFkAdresse().getPlz().getPlz().toString()).size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Plz");
```

```
        }
    }
    /**
     * Tests if Projekte are findable by Bauherr
     */
    @Test
    public void findProjektByBauherr() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            List<Projekt> listProjekt = projektdao.findAllProjekt();
            Projekt projekt = listProjekt.get(listProjekt.size() - 1);
            assertTrue(projektdao.findProjektByBauherr(
                    projekt.getFkBauherr().get(0).getNachname()).size() >
0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Bauherr");
        }
    }
    /**
     * Tests if Projekte are findable by Objekttyp
     */
    @Test
    public void findProjektByObjekttyp() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            List<Projekt> listProjekt = projektdao.findAllProjekt();
            Projekt projekt = listProjekt.get(listProjekt.size() - 1);
            assertTrue(projektdao.findProjektByObjekttyp(
                    projekt.getFkObjekttyp().getBezeichnung()).size() > 0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Objekttyp");
        }
    }

    /**
     * Test if Projekte are findable by Arbeitstyp
     */
    @Test
    public void findProjektByArbeitstyp() {
        try {
            ProjektDAO projektdao = new ProjektDAOImpl();
            List<Projekt> listProjekt = projektdao.findAllProjekt();
            Projekt projekt = listProjekt.get(listProjekt.size() - 1);
            assertTrue(projektdao.findProjektByArbeitstyp(
                    projekt.getFkArbeitstyp().getBezeichnung()).size() >
0);
        } catch (Exception e) {
            e.printStackTrace();
            fail("Couldn't find Projekt by Arbeitstyp");
        }
    }
///**
// * Tests if Projekte are findable by Subunternehmen
// */
//  @Test
//  public void findProjekByfindAllSubunternehmenProjekt() {
//      try {
//          ProjektDAO projektdao = new ProjektDAOImpl();
//          List<Projekt> listProjekt = projektdao.findAllProjekt();
//          Projekt projekt = listProjekt.get(listProjekt.size() - 1);
//          assertTrue(projektdao.findAllSubunternehmenProjekt(
```

```
//                  projekt.getFkProjektSuMitarbeiter().get(0)
//
.getFkMitarbeiter().getFkSubunternehmen()).size() > 0);
//      } catch (Exception e) {
//          e.printStackTrace();
//          fail("Couldn't find Projekt by findAllSubunternehmenProjekt");
//      }
//  }

}
```

AdresseRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.adresse;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Adresse;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * AdresseRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
 *
 */

public interface AdresseRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Adresse add(Adresse entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Adresse update(Adresse entity) throws RemoteException, Exception;

    /**
```

```
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Adresse entity) throws RemoteException, Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    Adresse findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Adresse> findAll() throws RemoteException;
}
```

AdresseROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.adresse;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.adresse.AdresseManager;
import ch.hsluw.mangelmanager.business.adresse.AdresseManagerImpl;
import ch.hsluw.mangelmanager.model.Adresse;

public class AdresseROImpl extends UnicastRemoteObject implements AdresseRO
{


    private static final long serialVersionUID = -4116064628937156721L;

    private AdresseManager adresseManager;

    public AdresseROImpl() throws RemoteException {
        adresseManager = new AdresseManagerImpl();
```

```
    }

    @Override
    public Adresse add(Adresse entity) throws RemoteException, Exception {
        return adresseManager.add(entity);
    }

    @Override
    public Adresse update(Adresse entity) throws RemoteException, Exception
{
        return adresseManager.update(entity);
    }

    @Override
    public void delete(Adresse entity) throws RemoteException, Exception {
        adresseManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        adresseManager.deleteById(id);
    }

    @Override
    public Adresse findById(Integer id) throws RemoteException {
        return adresseManager.findById(id);
    }

    @Override
    public List<Adresse> findAll() throws RemoteException {
        return adresseManager.findAll();
    }




}
```

ArbeitstypRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.arbeitstyp;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Arbeitstyp;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ArbeitstypRO zur Verfügung
 *
 * @version 1.0
```

```
 * @author sritz
 *
 */

public interface ArbeitstypRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Arbeitstyp add(Arbeitstyp entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Arbeitstyp update(Arbeitstyp entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Arbeitstyp entity) throws RemoteException, Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    Arbeitstyp findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Arbeitstyp> findAll() throws RemoteException;

    /**
     * Liefert die Liste mit Arbeitstypen für die übergebene Bezeichnung.
```

```
     *
     * @param bezeichnung
     * @return
     * @throws RemoteException
     */
    List<Arbeitstyp> findByBezeichnung(String bezeichnung)
            throws RemoteException;
}
```

ArbeitstypROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.arbeitstyp;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.arbeitstyp.ArbeitstypManager;
import ch.hsluw.mangelmanager.business.arbeitstyp.ArbeitstypManagerImpl;
import ch.hsluw.mangelmanager.model.Arbeitstyp;

public class ArbeitstypROImpl extends UnicastRemoteObject implements
ArbeitstypRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private ArbeitstypManager arbeitstypManager;

    public ArbeitstypROImpl() throws RemoteException {
        arbeitstypManager = new ArbeitstypManagerImpl();
    }

    @Override
    public Arbeitstyp add(Arbeitstyp entity) throws RemoteException,
Exception {
        return arbeitstypManager.add(entity);
    }

    @Override
    public Arbeitstyp update(Arbeitstyp entity) throws RemoteException,
Exception {
        return arbeitstypManager.update(entity);
    }

    @Override
    public void delete(Arbeitstyp entity) throws RemoteException, Exception
{
        arbeitstypManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        arbeitstypManager.deleteById(id);
    }

    @Override
```

```
    public Arbeitstyp findById(Integer id) throws RemoteException {
        return arbeitstypManager.findById(id);
    }

    @Override
    public List<Arbeitstyp> findAll() throws RemoteException {
        return arbeitstypManager.findAll();
    }

    @Override
    public List<Arbeitstyp> findByBezeichnung(String bezeichnung)
            throws RemoteException {
        return arbeitstypManager.findByBezeichnung(bezeichnung);
    }



}
```

BauherrRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.bauherr;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Bauherr;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * BauherrRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
 *
 */

public interface BauherrRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Bauherr add(Bauherr entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
```

```
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Bauherr update(Bauherr entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Bauherr entity) throws RemoteException, Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    Bauherr findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Bauherr> findAll() throws RemoteException;
}
```

BauherrROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.bauherr;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.bauherr.BauherrManager;
import ch.hsluw.mangelmanager.business.bauherr.BauherrManagerImpl;
import ch.hsluw.mangelmanager.model.Bauherr;

public class BauherrROImpl extends UnicastRemoteObject implements BauherrRO
{
```

```
    private static final long serialVersionUID = -4116064628937156721L;

    private BauherrManager bauherrManager;

    public BauherrROImpl() throws RemoteException {
        bauherrManager = new BauherrManagerImpl();
    }

    @Override
    public Bauherr add(Bauherr entity) throws RemoteException, Exception {
        return bauherrManager.add(entity);
    }

    @Override
    public Bauherr update(Bauherr entity) throws RemoteException, Exception
{
        return bauherrManager.update(entity);
    }

    @Override
    public void delete(Bauherr entity) throws RemoteException, Exception {
        bauherrManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        bauherrManager.deleteById(id);
    }

    @Override
    public Bauherr findById(Integer id) throws RemoteException {
        return bauherrManager.findById(id);
    }

    @Override
    public List<Bauherr> findAll() throws RemoteException {
        return bauherrManager.findAll();
    }




}
```

GuMitarbeiterRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.gumitarbeiter;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.GuMitarbeiter;
```

```
/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * GuMitarbeiterRO zur Verfügung
 *
 * @version 1.0
 * @author lkuendig
 *
 */

public interface GuMitarbeiterRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
     GuMitarbeiter add(GuMitarbeiter entity) throws RemoteException,
Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    GuMitarbeiter update(GuMitarbeiter entity) throws RemoteException,
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(GuMitarbeiter entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    GuMitarbeiter findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
```

```
     *
     * @return
     * @throws RemoteException
     */
    List<GuMitarbeiter> findAll() throws RemoteException;
}
```

GuMitarbeiterROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.gumitarbeiter;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.gumitarbeiter.GuMitarbeiterManager;
import
ch.hsluw.mangelmanager.business.gumitarbeiter.GuMitarbeiterManagerImpl;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;

public class GuMitarbeiterROImpl extends UnicastRemoteObject implements
GuMitarbeiterRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private GuMitarbeiterManager guMitarbeiterManager;

    public GuMitarbeiterROImpl() throws RemoteException {
        guMitarbeiterManager = new GuMitarbeiterManagerImpl();
    }

    @Override
    public GuMitarbeiter add(GuMitarbeiter entity) throws RemoteException,
Exception {
        return guMitarbeiterManager.add(entity);
    }

    @Override
    public GuMitarbeiter update(GuMitarbeiter entity) throws
RemoteException, Exception {
        return guMitarbeiterManager.update(entity);
    }

    @Override
    public void delete(GuMitarbeiter entity) throws RemoteException,
Exception {
        guMitarbeiterManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        guMitarbeiterManager.deleteById(id);
    }

    @Override
```

```
    public GuMitarbeiter findById(Integer id) throws RemoteException {
        return guMitarbeiterManager.findById(id);
    }

    @Override
    public List<GuMitarbeiter> findAll() throws RemoteException {
        return guMitarbeiterManager.findAll();
    }

}
```

LoginRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.login;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Login;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * LoginRO zur Verfügung
 *
 * @version 1.0
 * @author miten
 *
 */

public interface LoginRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Login add(Login entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Login update(Login entity) throws RemoteException, Exception;

    /**
```

```
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Login entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Login findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Login> findAll() throws RemoteException;

    Login findByName(String name) throws RemoteException;
}
```

LoginROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.login;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.login.LoginManager;
import ch.hsluw.mangelmanager.business.login.LoginManagerImpl;
import ch.hsluw.mangelmanager.model.Login;

public class LoginROImpl extends UnicastRemoteObject implements LoginRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private LoginManager loginManager;

    public LoginROImpl() throws RemoteException {
        loginManager = new LoginManagerImpl();
```

```
    }

    @Override
    public Login add(Login entity) throws RemoteException, Exception {
        return loginManager.add(entity);
    }

    @Override
    public Login update(Login entity) throws RemoteException, Exception {
        return loginManager.update(entity);
    }

    @Override
    public void delete(Login entity) throws RemoteException, Exception {
        loginManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        loginManager.deleteById(id);
    }

    @Override
    public Login findById(Integer id) throws RemoteException {
        return loginManager.findById(id);
    }

    @Override
    public List<Login> findAll() throws RemoteException {
        return loginManager.findAll();
    }

    @Override
    public Login findByName(String name) throws RemoteException {
        // TODO Auto-generated method stub
        return loginManager.findByName(name);
    }

}
```

MangelRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.mangel;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Date;
import java.util.List;

import ch.hsluw.mangelmanager.model.Mangel;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
```

```
 * MangelRO zur Verfügung
 *
 * @version 1.0
 * @author mmont
 *
 */

public interface MangelRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Mangel add(Mangel entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Mangel update(Mangel entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Mangel entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Mangel findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Mangel> findAll() throws RemoteException;
```

```
/**
 * Liefert die Liste mit Mängeln für die übergebene Bezeichnung.
 *
 * @param bezeichnung
 * @return
 * @throws RemoteException
 */
List<Mangel> findByBezeichnung(String bezeichnung)
        throws RemoteException;

/**
 * Liefert die Liste mit Mängeln für den übergebenen Projektstatus.
 *
 * @param mangelstatus
 * @return
 * @throws RemoteException
 */
List<Mangel> findByMangelstatus(String mangelstatus)
        throws RemoteException;

/**
 * Liefert die Liste mit Projekten für den übergebenen Ort.
 *
 * @param name
 * @return
 * @throws RemoteException
 */
List<Mangel> findByName(String name)
        throws RemoteException;

/**
 * Liefert die Liste mit Projekten für die übergebenen Postleitzahl.
 *
 * @param erfassungsZeit
 * @return
 * @throws RemoteException
 */
List<Mangel> findByErfassungsZeit(Date erfassungsZeit)
        throws RemoteException;

/**
 * Liefert die Liste mit Projekten für den übergebenen Bauherren.
 *
 * @param faelligkeitsDatum
 * @return
 * @throws RemoteException
 */
List<Mangel> findByFaelligkeitsDatum(Date faelligkeitsDatum)
        throws RemoteException;

/**
 * Liefert die Liste mit Projekten für den übergebenen Objekttyp.
 *
 * @param abschlussZeit
 * @return
 * @throws RemoteException
 */
List<Mangel> findByAbschlussZeit(Date abschlussZeit)
        throws RemoteException;

/**
 * Liefert alle Mängel von einem Projekt
 *
```

```
     * @param projekt
     * @return
     * @throws RemoteException
     */
    List<Mangel> findAllMangelProjekt(Integer projekt) throws
RemoteException;

}
```

MangelROImpl

```
/*
 * ZWECK: mangelManager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.mangel;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Date;
import java.util.List;

import ch.hsluw.mangelmanager.business.mangel.MangelManager;
import ch.hsluw.mangelmanager.business.mangel.MangelManagerImpl;
import ch.hsluw.mangelmanager.model.Mangel;

public class MangelROImpl extends UnicastRemoteObject implements MangelRO {

    private static final long serialVersionUID = -4116064628937156721L;

    private MangelManager mangelManager;

    public MangelROImpl() throws RemoteException {
        mangelManager = new MangelManagerImpl();
    }

    @Override
    public Mangel add(Mangel entity) throws RemoteException, Exception {
        return mangelManager.add(entity);
    }

    @Override
    public Mangel update(Mangel entity) throws RemoteException, Exception {
        return mangelManager.update(entity);
    }

    @Override
    public void delete(Mangel entity) throws RemoteException, Exception {
        mangelManager.delete(entity);
    }

    @Override
    public List<Mangel> findByErfassungsZeit(Date erfassungsZeit)
            throws RemoteException {
        return mangelManager.findByErfassungsZeit(erfassungsZeit);
    }

    @Override
    public List<Mangel> findByFaelligkeitsDatum(Date faelligkeitsDatum)
            throws RemoteException {
        return mangelManager.findByFaelligkeitsDatum(faelligkeitsDatum);
```

```
    }

    @Override
    public List<Mangel> findByAbschlussZeit(Date abschlussZeit)
            throws RemoteException {
        return mangelManager.findByAbschlussZeit(abschlussZeit);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        mangelManager.deleteById(id);

    }

    @Override
    public Mangel findById(Integer id) throws RemoteException {
        return mangelManager.findById(id);
    }

    @Override
    public List<Mangel> findAll() throws RemoteException {
        return mangelManager.findAll();
    }

    @Override
    public List<Mangel> findByBezeichnung(String bezeichnung)
            throws RemoteException {
        return mangelManager.findByBezeichnung(bezeichnung);
    }

    @Override
    public List<Mangel> findByMangelstatus(String mangelstatus)
            throws RemoteException {
        return mangelManager.findByMangelstatus(mangelstatus);
    }

    @Override
    public List<Mangel> findByName(String name) throws RemoteException {
        return mangelManager.findByName(name);
    }

    @Override
    public List<Mangel> findAllMangelProjekt(Integer projekt) throws
RemoteException {
        return mangelManager.findAllMangelProjekt(projekt);
    }

}
```

MangelstatusRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.mangelstatus;


import java.rmi.Remote;
import java.rmi.RemoteException;
```

```java
import java.util.List;

import ch.hsluw.mangelmanager.model.Mangelstatus;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MangelRO zur Verfügung
 *
 * @version 1.0
 * @author mmont
 *
 */

public interface MangelstatusRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Mangelstatus add(Mangelstatus entity) throws RemoteException,
Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Mangelstatus update(Mangelstatus entity) throws RemoteException,
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Mangelstatus entity) throws RemoteException, Exception;

    List<Mangelstatus> findAllMangelStatus() throws RemoteException,
Exception;



}
```

MangelstatusROImpl

```java
/*
 * ZWECK: mangelManager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
package ch.hsluw.mangelmanager.rmi.mangelstatus;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.mangelstatus.MangelstatusManager;
import
ch.hsluw.mangelmanager.business.mangelstatus.MangelstatusManagerImpl;
import ch.hsluw.mangelmanager.model.Mangelstatus;


public class MangelstatusROImpl extends UnicastRemoteObject implements
MangelstatusRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private MangelstatusManager mangelstatusManager;

    public MangelstatusROImpl() throws RemoteException {
        mangelstatusManager = new MangelstatusManagerImpl();
    }

    @Override
    public Mangelstatus add(Mangelstatus entity) throws RemoteException,
Exception {
        return mangelstatusManager.add(entity);
    }

    @Override
    public Mangelstatus update(Mangelstatus entity) throws RemoteException,
Exception {
        return mangelstatusManager.update(entity);
    }

    @Override
    public void delete(Mangelstatus entity) throws RemoteException,
Exception {
        mangelstatusManager.delete(entity);
    }

    @Override
    public List<Mangelstatus> findAllMangelStatus() throws RemoteException,
            Exception {
        // TODO Auto-generated method stub
        return mangelstatusManager.getAllMangelstatus();
    }

}
```

MeldungRO

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.meldung;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
```

```java
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MeldungRO zur Verfügung
 *
 * @version 1.0
 * @author cdemir
 *
 */

public interface MeldungRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Meldung add(Meldung entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Meldung update(Meldung entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Meldung entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Meldung findById(Integer id) throws RemoteException;
```

```
    /**
     * Liefert alle Entity-Meldungen zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Meldung> findAll() throws RemoteException;

    List<Meldung> findAllMeldungByMangel(Mangel mangel) throws
RemoteException;

}
```

MeldungROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.meldung;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.meldung.MeldungManager;
import ch.hsluw.mangelmanager.business.meldung.MeldungManagerImpl;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Meldung;



public class MeldungROImpl extends UnicastRemoteObject implements MeldungRO
{


    private static final long serialVersionUID = -4240817211813159411L;

    private MeldungManager meldungManager;

    public MeldungROImpl() throws RemoteException {
        meldungManager = new MeldungManagerImpl();
    }

    @Override
    public Meldung add(Meldung entity) throws RemoteException, Exception {
        return meldungManager.add(entity);
    }

    @Override
    public Meldung update(Meldung entity) throws RemoteException, Exception
{
        return meldungManager.update(entity);
    }

    @Override
    public void delete(Meldung entity) throws RemoteException, Exception {
        meldungManager.delete(entity);
    }

    @Override
```

```
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub

    }

    @Override
    public Meldung findById(Integer id) throws RemoteException {
        // TODO Auto-generated method stub
        return meldungManager.findById(id);
    }

    @Override
    public List<Meldung> findAll() throws RemoteException {
        // TODO Auto-generated method stub
        return meldungManager.findAll();
    }

    @Override
    public List<Meldung> findAllMeldungByMangel(Mangel mangel) {
        // TODO Auto-generated method stub
        return meldungManager.findAllMeldungByMangel(mangel);
    }

}
```

MeldungstypRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.meldungstyp;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * MeldungstypRO zur Verfügung
 *
 * @version 1.0
 * @author cdemir
 *
 */

public interface MeldungstypRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
```

```
    Meldungstyp add(Meldungstyp entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Meldungstyp update(Meldungstyp entity) throws RemoteException,
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Meldungstyp entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Meldung findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Meldungen zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Meldungstyp> findAll() throws RemoteException;

}
```

MeldungstypROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekomponenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.meldungstyp;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;
```

```
import ch.hsluw.mangelmanager.business.meldungstyp.MeldungstypManager;
import ch.hsluw.mangelmanager.business.meldungstyp.MeldungstypManagerImpl;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;



public class MeldungstypROImpl extends UnicastRemoteObject implements
MeldungstypRO {


    private static final long serialVersionUID = -8523214358712447146L;

    private MeldungstypManager meldungstypManager;

    public MeldungstypROImpl() throws RemoteException {
        meldungstypManager = new MeldungstypManagerImpl();
    }

    @Override
    public Meldungstyp add(Meldungstyp entity) throws RemoteException,
Exception {
        return meldungstypManager.add(entity);
    }

    @Override
    public Meldungstyp update(Meldungstyp entity) throws RemoteException,
Exception {
        return meldungstypManager.update(entity);
    }

    @Override
    public void delete(Meldungstyp entity) throws RemoteException,
Exception {
        meldungstypManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        // TODO Auto-generated method stub

    }

    @Override
    public Meldung findById(Integer id) throws RemoteException {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public List<Meldungstyp> findAll() throws RemoteException {
        // TODO Auto-generated method stub
        return meldungstypManager.findAll();
    }

}
```

ObjekttypRO

```
/*
 * ZWECK: Mangelmanager
```

```java
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.objekttyp;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Objekttyp;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ObjekttypRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
 *
 */

public interface ObjekttypRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Objekttyp add(Objekttyp entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Objekttyp update(Objekttyp entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Objekttyp entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
```

```
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Objekttyp findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Objekttyp> findAll() throws RemoteException;

    /**
     * Liefert die Liste mit Objekttypen für die übergebene Bezeichnung.
     *
     * @param bezeichnung
     * @return
     * @throws RemoteException
     */
    List<Objekttyp> findByBezeichnung(String bezeichnung)
            throws RemoteException;
}
```

ObjekttypROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.objekttyp;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.objekttyp.ObjekttypManager;
import ch.hsluw.mangelmanager.business.objekttyp.ObjekttypManagerImpl;
import ch.hsluw.mangelmanager.model.Objekttyp;

public class ObjekttypROImpl extends UnicastRemoteObject implements
ObjekttypRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private ObjekttypManager objekttypManager;

    public ObjekttypROImpl() throws RemoteException {
        objekttypManager = new ObjekttypManagerImpl();
    }

    @Override
    public Objekttyp add(Objekttyp entity) throws RemoteException,
Exception {
        return objekttypManager.add(entity);
    }
```

```
    @Override
    public Objekttyp update(Objekttyp entity) throws RemoteException,
Exception {
        return objekttypManager.update(entity);
    }

    @Override
    public void delete(Objekttyp entity) throws RemoteException, Exception
{
        objekttypManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        objekttypManager.deleteById(id);
    }

    @Override
    public Objekttyp findById(Integer id) throws RemoteException {
        return objekttypManager.findById(id);
    }

    @Override
    public List<Objekttyp> findAll() throws RemoteException {
        return objekttypManager.findAll();
    }

    @Override
    public List<Objekttyp> findByBezeichnung(String bezeichnung)
            throws RemoteException {
        return objekttypManager.findByBezeichnung(bezeichnung);
    }



}
```

PersonRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.person;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Person;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * PersonRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
```

```
 *
 */

public interface PersonRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Person add(Person entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Person update(Person entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Person entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Person findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Person> findAll() throws RemoteException;
}
```

PersonROImpl

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.person;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.person.PersonManager;
import ch.hsluw.mangelmanager.business.person.PersonManagerImpl;
import ch.hsluw.mangelmanager.model.Person;

public class PersonROImpl extends UnicastRemoteObject implements PersonRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private PersonManager personManager;

    public PersonROImpl() throws RemoteException {
        personManager = new PersonManagerImpl();
    }

    @Override
    public Person add(Person entity) throws RemoteException, Exception {
        return personManager.add(entity);
    }

    @Override
    public Person update(Person entity) throws RemoteException, Exception {
        return personManager.update(entity);
    }

    @Override
    public void delete(Person entity) throws RemoteException, Exception {
        personManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        personManager.deleteById(id);
    }

    @Override
    public Person findById(Integer id) throws RemoteException {
        return personManager.findById(id);
    }

    @Override
    public List<Person> findAll() throws RemoteException {
        return personManager.findAll();
    }

}
```

PlzRO

```java
/*
```

```java
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.plz;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Plz;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * PlzRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
 *
 */

public interface PlzRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Plz add(Plz entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Plz update(Plz entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Plz entity) throws RemoteException, Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
```

```
    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    Plz findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Plz> findAll() throws RemoteException;
}
```

PlzROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.plz;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.plz.PlzManager;
import ch.hsluw.mangelmanager.business.plz.PlzManagerImpl;
import ch.hsluw.mangelmanager.model.Plz;

public class PlzROImpl extends UnicastRemoteObject implements PlzRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private PlzManager adresseManager;

    public PlzROImpl() throws RemoteException {
        adresseManager = new PlzManagerImpl();
    }

    @Override
    public Plz add(Plz entity) throws RemoteException, Exception {
        return adresseManager.add(entity);
    }

    @Override
    public Plz update(Plz entity) throws RemoteException, Exception {
        return adresseManager.update(entity);
    }

    @Override
    public void delete(Plz entity) throws RemoteException, Exception {
        adresseManager.delete(entity);
    }
```

```
    @Override
    public void deleteById(Integer id) throws Exception {
        adresseManager.deleteById(id);
    }

    @Override
    public Plz findById(Integer id) throws RemoteException {
        return adresseManager.findById(id);
    }

    @Override
    public List<Plz> findAll() throws RemoteException {
        return adresseManager.findAll();
    }




}
```

ProjektRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projekt;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Projekt;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
 *
 */

public interface ProjektRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Projekt add(Projekt entity) throws RemoteException, Exception;

    /**
```

```
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Projekt update(Projekt entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Projekt entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Projekt findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Projekt> findAll() throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für die übergebene Bezeichnung.
     *
     * @param bezeichnung
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByBezeichnung(String bezeichnung)
            throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für den übergebenen Projektstatus.
     *
     * @param projektstatus
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByProjektstatus(String projektstatus)
            throws RemoteException;
```

```
    /**
     * Liefert die Liste mit Projekten für den übergebenen Ort.
     *
     * @param ort
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByOrt(String ort)
            throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für die übergebenen Postleitzahl.
     *
     * @param plz
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByPlz(String plz)
            throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für den übergebenen Bauherren.
     *
     * @param bauherr
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByBauherr(String bauherr)
            throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für den übergebenen Objekttyp.
     *
     * @param objekttyp
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByObjekttyp(String objekttyp)
            throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für den übergebenen Arbeitstyp.
     *
     * @param arbeitstyp
     * @return
     * @throws RemoteException
     */
    List<Projekt> findByArbeitstyp(String arbeitstyp)
            throws RemoteException;

    /**
     * Liefert die Liste mit Projekten für den übergebenen Zeitrahmen.
     *
     * @param fromDatum
     * @param endDatum
     * @return
     * @throws RemoteException
     */

    List<Projekt> findAllSubunternehmenProjekt(Integer subunternehmen2)
throws RemoteException;
```

```
    List<Projekt> findProjektbyPerson(Person person) throws
RemoteException;


}
```

ProjektROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projekt;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.projekt.ProjektManager;
import ch.hsluw.mangelmanager.business.projekt.ProjektManagerImpl;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Projekt;

public class ProjektROImpl extends UnicastRemoteObject implements ProjektRO
{


    private static final long serialVersionUID = -4116064628937156721L;

    private ProjektManager projektManager;

    public ProjektROImpl() throws RemoteException {
        projektManager = new ProjektManagerImpl();
    }

    @Override
    public Projekt add(Projekt entity) throws RemoteException, Exception {
        return projektManager.add(entity);
    }

    @Override
    public Projekt update(Projekt entity) throws RemoteException, Exception
{
        return projektManager.update(entity);
    }

    @Override
    public void delete(Projekt entity) throws RemoteException, Exception {
        projektManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        projektManager.deleteById(id);
    }

    @Override
    public Projekt findById(Integer id) throws RemoteException {
        return projektManager.findById(id);
    }
```

```
    @Override
    public List<Projekt> findAll() throws RemoteException {
        return projektManager.findAll();
    }

    @Override
    public List<Projekt> findByBezeichnung(String bezeichnung)
            throws RemoteException {
        return projektManager.findByBezeichnung(bezeichnung);
    }

    @Override
    public List<Projekt> findByProjektstatus(String projektstatus)
            throws RemoteException {
        return projektManager.findByProjektstatus(projektstatus);
    }

    @Override
    public List<Projekt> findByOrt(String ort)
            throws RemoteException {
        return projektManager.findByOrt(ort);
    }

    @Override
    public List<Projekt> findByPlz(String plz)
            throws RemoteException {
        return projektManager.findByPlz(plz);
    }

    @Override
    public List<Projekt> findByBauherr(String bauherr)
            throws RemoteException {
        return projektManager.findByBauherr(bauherr);
    }

    @Override
    public List<Projekt> findByObjekttyp(String objekttyp)
            throws RemoteException {
        return projektManager.findByObjekttyp(objekttyp);
    }

    @Override
    public List<Projekt> findByArbeitstyp(String arbeitstyp)
            throws RemoteException {
        return projektManager.findByArbeitstyp(arbeitstyp);
    }


    @Override
    public List<Projekt> findAllSubunternehmenProjekt(Integer
subunternehmen2) throws RemoteException {
        return
projektManager.findAllSubunternehmenProjekt(subunternehmen2);
    }

    @Override
    public List<Projekt> findProjektbyPerson(Person person)
            throws RemoteException {
        // TODO Auto-generated method stub
        return projektManager.findProjektbyPerson(person);
    }
```

```
}
```

ProjektGuMitarbeiterRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projektgumitarbeiter;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektGuMitarbeiterRO zur Verfügung
 *
 * @version 1.0
 * @author lkuendig
 *
 */

public interface ProjektGuMitarbeiterRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    ProjektGuMitarbeiter add(ProjektGuMitarbeiter entity) throws
RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    ProjektGuMitarbeiter update(ProjektGuMitarbeiter entity) throws
RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
```

```
    void delete(ProjektGuMitarbeiter entity) throws RemoteException,
Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception;

    /**
     * Liefert die Entity für die übergebenen Werte zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    ProjektGuMitarbeiter findById(Integer idProjekt, Integer idMitarbeiter)
throws RemoteException, Exception;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<ProjektGuMitarbeiter> findAll() throws RemoteException;

    List<ProjektGuMitarbeiter> findAllBauleiterByProjekt(Projekt projekt2)
throws RemoteException, Exception;
}
```

ProjektGuMitarbeiterROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projektgumitarbeiter;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import
ch.hsluw.mangelmanager.business.projektgumitarbeiter.ProjektGuMitarbeiterMa
nager;
import
ch.hsluw.mangelmanager.business.projektgumitarbeiter.ProjektGuMitarbeiterMa
nagerImpl;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;

public class ProjektGuMitarbeiterROImpl extends UnicastRemoteObject
implements ProjektGuMitarbeiterRO {
```

```
    private static final long serialVersionUID = -4116064628937156721L;

    private ProjektGuMitarbeiterManager projektGuMitarbeiterManager;

    public ProjektGuMitarbeiterROImpl() throws RemoteException {
        projektGuMitarbeiterManager = new
ProjektGuMitarbeiterManagerImpl();
    }

    @Override
    public ProjektGuMitarbeiter add(ProjektGuMitarbeiter entity) throws
RemoteException, Exception {
        return projektGuMitarbeiterManager.add(entity);
    }

    @Override
    public ProjektGuMitarbeiter update(ProjektGuMitarbeiter entity) throws
RemoteException, Exception {
        return projektGuMitarbeiterManager.update(entity);
    }

    @Override
    public void delete(ProjektGuMitarbeiter entity) throws RemoteException,
Exception {
        projektGuMitarbeiterManager.delete(entity);
    }

    @Override
    public void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception {
        projektGuMitarbeiterManager.deleteById(idProjekt, idMitarbeiter);
    }

    @Override
    public ProjektGuMitarbeiter findById(Integer idProjekt, Integer
idMitarbeiter) throws Exception {
        return projektGuMitarbeiterManager.findById(idProjekt,
idMitarbeiter);
    }

    @Override
    public List<ProjektGuMitarbeiter> findAll() throws RemoteException {
        return projektGuMitarbeiterManager.findAll();
    }

    @Override
    public List<ProjektGuMitarbeiter> findAllBauleiterByProjekt(Projekt
projekt2)  throws RemoteException, Exception {
        // TODO Auto-generated method stub
        return
projektGuMitarbeiterManager.findAllBauleiterByProjekt(projekt2);
    }

}
```

ProjektstatusRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */
```

```java
package ch.hsluw.mangelmanager.rmi.projektstatus;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Projektstatus;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektstatusRO zur Verfügung
 *
 * @version 1.0
 * @author sritz
 *
 */

public interface ProjektstatusRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Projektstatus add(Projektstatus entity) throws RemoteException,
Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Projektstatus update(Projektstatus entity) throws RemoteException,
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Projektstatus entity) throws RemoteException, Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */

    /**
```

```
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    Projektstatus findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Projektstatus> findAll() throws RemoteException;
}
```

ProjektstatusROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projektstatus;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.projektstatus.ProjektstatusManager;
import
ch.hsluw.mangelmanager.business.projektstatus.ProjektstatusManagerImpl;
import ch.hsluw.mangelmanager.model.Projektstatus;

public class ProjektstatusROImpl extends UnicastRemoteObject implements
ProjektstatusRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private ProjektstatusManager projektstatusManager;

    public ProjektstatusROImpl() throws RemoteException {
        projektstatusManager = new ProjektstatusManagerImpl();
    }

    @Override
    public Projektstatus add(Projektstatus entity) throws RemoteException,
Exception {
        return projektstatusManager.add(entity);
    }

    @Override
    public Projektstatus update(Projektstatus entity) throws
RemoteException, Exception {
        return projektstatusManager.update(entity);
    }

    @Override
```

```
    public void delete(Projektstatus entity) throws RemoteException,
Exception {
        projektstatusManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        projektstatusManager.deleteById(id);
    }

    @Override
    public Projektstatus findById(Integer id) throws RemoteException {
        return projektstatusManager.findById(id);
    }

    @Override
    public List<Projektstatus> findAll() throws RemoteException {
        return projektstatusManager.findAll();
    }




}
```

ProjektSuMitarbeiterRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projektsumitarbeiter;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * ProjektSuMitarbeiterRO zur Verfügung
 *
 * @version 1.0
 * @author lkuendig
 *
 */

public interface ProjektSuMitarbeiterRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
```

```
    ProjektSuMitarbeiter add(ProjektSuMitarbeiter entity) throws
RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    ProjektSuMitarbeiter update(ProjektSuMitarbeiter entity) throws
RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(ProjektSuMitarbeiter entity) throws RemoteException,
Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception;

    /**
     * Liefert die Entity für die übergebenen Werte zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    ProjektSuMitarbeiter findById(Integer idProjekt, Integer idMitarbeiter)
throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<ProjektSuMitarbeiter> findAll() throws RemoteException;
}
```

ProjektSuMitarbeiterROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.projektsumitarbeiter;
```

```java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import
ch.hsluw.mangelmanager.business.projektsumitarbeiter.ProjektSuMitarbeiterMa
nager;
import
ch.hsluw.mangelmanager.business.projektsumitarbeiter.ProjektSuMitarbeiterMa
nagerImpl;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;

public class ProjektSuMitarbeiterROImpl extends UnicastRemoteObject
implements ProjektSuMitarbeiterRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private ProjektSuMitarbeiterManager projektSuMitarbeiterManager;

    public ProjektSuMitarbeiterROImpl() throws RemoteException {
        projektSuMitarbeiterManager = new
ProjektSuMitarbeiterManagerImpl();
    }

    @Override
    public ProjektSuMitarbeiter add(ProjektSuMitarbeiter entity) throws
RemoteException, Exception {
        return projektSuMitarbeiterManager.add(entity);
    }

    @Override
    public ProjektSuMitarbeiter update(ProjektSuMitarbeiter entity) throws
RemoteException, Exception {
        return projektSuMitarbeiterManager.update(entity);
    }

    @Override
    public void delete(ProjektSuMitarbeiter entity) throws RemoteException,
Exception {
        projektSuMitarbeiterManager.delete(entity);
    }

    @Override
    public void deleteById(Integer idProjekt, Integer idMitarbeiter) throws
Exception {
        projektSuMitarbeiterManager.deleteById(idProjekt, idMitarbeiter);
    }

    @Override
    public ProjektSuMitarbeiter findById(Integer idProjekt, Integer
idMitarbeiter) throws RemoteException {
        return projektSuMitarbeiterManager.findById(idProjekt,
idMitarbeiter);
    }

    @Override
    public List<ProjektSuMitarbeiter> findAll() throws RemoteException {
        return projektSuMitarbeiterManager.findAll();
    }

}
```

RolleRO

```java
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.rolle;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.Rolle;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * RolleRO zur Verfügung
 *
 * @version 1.0
 * @author miten
 *
 */

public interface RolleRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Rolle add(Rolle entity) throws RemoteException, Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    Rolle update(Rolle entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Rolle entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
```

```
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Rolle findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Rolle> findAll() throws RemoteException;
}
```

RolleROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.rolle;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import ch.hsluw.mangelmanager.business.rolle.RolleManager;
import ch.hsluw.mangelmanager.business.rolle.RolleManagerImpl;
import ch.hsluw.mangelmanager.model.Rolle;

public class RolleROImpl extends UnicastRemoteObject implements RolleRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private RolleManager rolleManager;

    public RolleROImpl() throws RemoteException {
        rolleManager = new RolleManagerImpl();
    }

    @Override
    public Rolle add(Rolle entity) throws RemoteException, Exception {
        return rolleManager.add(entity);
    }

    @Override
    public Rolle update(Rolle entity) throws RemoteException, Exception {
        return rolleManager.update(entity);
    }

    @Override
```

```
    public void delete(Rolle entity) throws RemoteException, Exception {
        rolleManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        rolleManager.deleteById(id);
    }

    @Override
    public Rolle findById(Integer id) throws RemoteException {
        return rolleManager.findById(id);
    }

    @Override
    public List<Rolle> findAll() throws RemoteException {
        return rolleManager.findAll();
    }
}
```

RMIServer

```
package ch.hsluw.mangelmanager.rmi.server;

import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Properties;

import javax.swing.JOptionPane;

import ch.hsluw.mangelmanager.rmi.adresse.AdresseRO;
import ch.hsluw.mangelmanager.rmi.adresse.AdresseROImpl;
import ch.hsluw.mangelmanager.rmi.arbeitstyp.ArbeitstypRO;
import ch.hsluw.mangelmanager.rmi.arbeitstyp.ArbeitstypROImpl;
import ch.hsluw.mangelmanager.rmi.bauherr.BauherrRO;
import ch.hsluw.mangelmanager.rmi.bauherr.BauherrROImpl;
import ch.hsluw.mangelmanager.rmi.gumitarbeiter.GuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.gumitarbeiter.GuMitarbeiterROImpl;
import ch.hsluw.mangelmanager.rmi.login.LoginRO;
import ch.hsluw.mangelmanager.rmi.login.LoginROImpl;
import ch.hsluw.mangelmanager.rmi.mangel.MangelRO;
import ch.hsluw.mangelmanager.rmi.mangel.MangelROImpl;
import ch.hsluw.mangelmanager.rmi.mangelstatus.MangelstatusRO;
import ch.hsluw.mangelmanager.rmi.mangelstatus.MangelstatusROImpl;
import ch.hsluw.mangelmanager.rmi.meldung.MeldungRO;
import ch.hsluw.mangelmanager.rmi.meldung.MeldungROImpl;
import ch.hsluw.mangelmanager.rmi.meldungstyp.MeldungstypRO;
import ch.hsluw.mangelmanager.rmi.meldungstyp.MeldungstypROImpl;
import ch.hsluw.mangelmanager.rmi.objekttyp.ObjekttypRO;
import ch.hsluw.mangelmanager.rmi.objekttyp.ObjekttypROImpl;
import ch.hsluw.mangelmanager.rmi.person.PersonRO;
import ch.hsluw.mangelmanager.rmi.person.PersonROImpl;
import ch.hsluw.mangelmanager.rmi.plz.PlzRO;
import ch.hsluw.mangelmanager.rmi.plz.PlzROImpl;
import ch.hsluw.mangelmanager.rmi.projekt.ProjektRO;
import ch.hsluw.mangelmanager.rmi.projekt.ProjektROImpl;
import
ch.hsluw.mangelmanager.rmi.projektgumitarbeiter.ProjektGuMitarbeiterROImpl;
import ch.hsluw.mangelmanager.rmi.projektstatus.ProjektstatusRO;
```

```
import ch.hsluw.mangelmanager.rmi.projektstatus.ProjektstatusROImpl;
import
ch.hsluw.mangelmanager.rmi.projektsumitarbeiter.ProjektSuMitarbeiterRO;
import
ch.hsluw.mangelmanager.rmi.projektsumitarbeiter.ProjektSuMitarbeiterROImpl;
import ch.hsluw.mangelmanager.rmi.rolle.RolleRO;
import ch.hsluw.mangelmanager.rmi.rolle.RolleROImpl;
import ch.hsluw.mangelmanager.rmi.subunternehmen.SubunternehmenRO;
import ch.hsluw.mangelmanager.rmi.subunternehmen.SubunternehmenROImpl;
import ch.hsluw.mangelmanager.rmi.sumitarbeiter.SuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.sumitarbeiter.SuMitarbeiterROImpl;


public class RMIServer {

    public static void main(String[] args) {

        /*
         * Port und Host-IP
         */
        int port;
        String hostIp;

        /* Properties laden */
        Properties props = new Properties();

        InputStream is = RMIServer.class.getClassLoader()
                .getResourceAsStream("rmi.properties");

        try {
            props.load(is);
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        hostIp = props.getProperty("rmi.host_ip");
        port = Integer.parseInt(props.getProperty("rmi.port"));

        Registry registry = null;

        try {

            InetAddress.getLocalHost();

            /*
             * Die NIC-IP nach 'aussen' kommunizieren (falls die
Namensauflösung
             * probleme bereiten sollte)
             */
            System.setProperty("java.rmi.server.hostname", hostIp);

            // Registry starten
            registry = LocateRegistry.createRegistry(port);

            if (registry != null) {

                // Entfernte Objekte erstellen
                PersonRO personRO = new PersonROImpl();
                ProjektRO projektRO = new ProjektROImpl();
                SubunternehmenRO subunternehmenRO = new
SubunternehmenROImpl();
                MangelRO mangelRO = new MangelROImpl();
```

```
                MeldungRO meldungRO = new MeldungROImpl();
                PlzRO plzRO = new PlzROImpl();
                AdresseRO adresseRO = new AdresseROImpl();
                ObjekttypRO objekttypRO = new ObjekttypROImpl();
                ArbeitstypRO arbeitstypRO = new ArbeitstypROImpl();
                MangelstatusRO mangelstatusRO = new MangelstatusROImpl();
                MeldungstypRO meldungstypRO = new MeldungstypROImpl();
                ProjektGuMitarbeiterROImpl projektGuMitarbeiterRO = new
ProjektGuMitarbeiterROImpl();
                LoginRO loginRO = new LoginROImpl();
                ProjektSuMitarbeiterRO projektSuMitarbeiterRO = new
ProjektSuMitarbeiterROImpl();
                BauherrRO bauherrRO = new BauherrROImpl();
                GuMitarbeiterRO guMitarbeiterRO = new
GuMitarbeiterROImpl();
                SuMitarbeiterRO suMitarbeiterRO = new
SuMitarbeiterROImpl();
                ProjektstatusRO projektstatusRO = new
ProjektstatusROImpl();
                RolleRO rolleRO = new RolleROImpl();

                registry.rebind("personRO", personRO);
                registry.rebind("projektRO", projektRO);
                registry.rebind("subunternehmenRO", subunternehmenRO);
                registry.rebind("mangelRO", mangelRO);
                registry.rebind("meldungRO", meldungRO);
                registry.rebind("plzRO", plzRO);
                registry.rebind("adresseRO", adresseRO);
                registry.rebind("objekttypRO", objekttypRO);
                registry.rebind("arbeitstypRO", arbeitstypRO);
                registry.rebind("mangelstatusRO", mangelstatusRO);
                registry.rebind("meldungstypRO", meldungstypRO);
                registry.rebind("loginRO", loginRO);
                registry.rebind("projektSuMitarbeiterRO",
projektSuMitarbeiterRO);
                registry.rebind("projektGuMitarbeiterRO",
projektGuMitarbeiterRO);
                registry.rebind("bauherrRO", bauherrRO);
                registry.rebind("guMitarbeiterRO", guMitarbeiterRO);
                registry.rebind("projektstatusRO", projektstatusRO);
                registry.rebind("suMitarbeiterRO", suMitarbeiterRO);
                registry.rebind("rolleRO", rolleRO);


                String msg = "RMI-Server ist bereit für Client-
Anfragen.\n\n"
                        + "Server herunterfahren?";
                JOptionPane.showMessageDialog(null, msg, hostIp + ":" +
port ,
                        JOptionPane.QUESTION_MESSAGE);

                registry.unbind("personRO");
                registry.unbind("projektRO");
                registry.unbind("subunternehmenRO");
                registry.unbind("mangelRO");
                registry.unbind("meldungRO");
                registry.unbind("plzRO");
                registry.unbind("adresseRO");
                registry.unbind("objekttypRO");
                registry.unbind("arbeitstypRO");
                registry.unbind("mangelstatusRO");
                registry.unbind("meldungstypRO");
                registry.unbind("projektGuMitarbeiterRO");
```

```
                registry.unbind("loginRO");
                registry.unbind("projektSuMitarbeiterRO");
                registry.unbind("bauherrRO");
                registry.unbind("guMitarbeiterRO");
                registry.unbind("projektstatusRO");
                registry.unbind("suMitarbeiterRO");
                registry.unbind("rolleRO");

                System.out.println("RMI Server wird heruntergefahren!\n");

                System.exit(0);
            } else {
                System.out
                        .println("Entferntes Objekt konnte nicht exportiert
werden!");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }



}
```

SubunternehmenRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.subunternehmen;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * SubunternehmenRO zur Verfügung
 *
 * @version 1.0
 * @author lkuendig
 *
 */

public interface SubunternehmenRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
```

```
     */
    Subunternehmen add(Subunternehmen entity) throws RemoteException,
Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    void update(Subunternehmen entity) throws RemoteException, Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(Subunternehmen entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    Subunternehmen findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<Subunternehmen> findAll() throws RemoteException;

    String findAllProjekte(int subunternehmen) throws RemoteException;

    List<SuMitarbeiter> findAllSubunternehmenMitarbeiter(Subunternehmen
subunternehmen) throws RemoteException;

    List<Subunternehmen> findAllSubunternehmenByProjekt(Integer projekt2)
throws RemoteException;

}
```

SubunternehmenROImpl

```
/*
```

```
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.subunternehmen;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

import
ch.hsluw.mangelmanager.business.subunternehmen.SubunternehmenManager;
import
ch.hsluw.mangelmanager.business.subunternehmen.SubunternehmenManagerImpl;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

public class SubunternehmenROImpl extends UnicastRemoteObject implements
SubunternehmenRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private SubunternehmenManager subunternehmenManager;

    public SubunternehmenROImpl() throws RemoteException {
        subunternehmenManager = new SubunternehmenManagerImpl();
    }

    @Override
    public Subunternehmen add(Subunternehmen entity) throws
RemoteException, Exception {
        return subunternehmenManager.add(entity);
    }

    @Override
    public void update(Subunternehmen entity) throws RemoteException,
Exception {
        subunternehmenManager.update(entity);
    }

    @Override
    public void delete(Subunternehmen entity) throws RemoteException,
Exception {
        subunternehmenManager.delete(entity);
    }

    @Override
    public void deleteById(Integer id) throws Exception {
        subunternehmenManager.deleteById(id);
    }

    @Override
    public Subunternehmen findById(Integer id) throws RemoteException {
        return subunternehmenManager.findById(id);
    }

    @Override
    public List<Subunternehmen> findAll() throws RemoteException {
        return subunternehmenManager.findAll();
    }

    @Override
```

```
    public String findAllProjekte(int subunternehmen) throws
RemoteException {
        return subunternehmenManager.findAllProjekte(subunternehmen);
    }

    @Override
    public List<SuMitarbeiter>
findAllSubunternehmenMitarbeiter(Subunternehmen subunternehmen) throws
RemoteException {
        return
subunternehmenManager.findAllSubunternehmenMitarbeiter(subunternehmen);
    }

    @Override
    public List<Subunternehmen> findAllSubunternehmenByProjekt(Integer
projekt2) throws RemoteException {
        // TODO Auto-generated method stub
        return
subunternehmenManager.findAllSubunternehmenByProjekt(projekt2);
    }


}
```

SuMitarbeiterRO

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.sumitarbeiter;


import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import ch.hsluw.mangelmanager.model.SuMitarbeiter;



/**
 * Diese Klasse stellt die Implementierung von Methoden der Schnittstelle
 * SuMitarbeiterRO zur Verfügung
 *
 * @version 1.0
 * @author lkuendig
 *
 */

public interface SuMitarbeiterRO extends Remote {

    /**
     * Speichert die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
```

```
    SuMitarbeiter add(SuMitarbeiter entity) throws RemoteException,
Exception;

    /**
     * Updatet die übergebene Entity.
     *
     * @param entity
     * @return
     * @throws RemoteException
     * @throws Exception
     */
    SuMitarbeiter update(SuMitarbeiter entity) throws RemoteException,
Exception;

    /**
     * Löscht die übergebene Entity.
     *
     * @param entity
     * @throws RemoteException
     * @throws Exception
     */
    void delete(SuMitarbeiter entity) throws RemoteException, Exception;

    /**
     * Löscht die Entity mit der übergebenen Id.
     *
     * @param entity
     * @throws Exception
     */
    void deleteById(Integer id) throws Exception;

    /**
     * Liefert die Entity für den übergebenen Id-Wert zurück.
     *
     * @param id
     * @return
     * @throws RemoteException
     */
    SuMitarbeiter findById(Integer id) throws RemoteException;

    /**
     * Liefert alle Entity-Objekte zurück.
     *
     * @return
     * @throws RemoteException
     */
    List<SuMitarbeiter> findAll() throws RemoteException;
}
```

SuMitarbeiterROImpl

```
/*
 * ZWECK: Mangelmanager
 * MODUL: Softwarekompomenten, HSLU-Wirtschaft
 */

package ch.hsluw.mangelmanager.rmi.sumitarbeiter;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;
```

```
import ch.hsluw.mangelmanager.business.sumitarbeiter.SuMitarbeiterManager;
import
ch.hsluw.mangelmanager.business.sumitarbeiter.SuMitarbeiterManagerImpl;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;

public class SuMitarbeiterROImpl extends UnicastRemoteObject implements
SuMitarbeiterRO {


    private static final long serialVersionUID = -4116064628937156721L;

    private SuMitarbeiterManager suMitarbeiterManager;

    public SuMitarbeiterROImpl() throws RemoteException {
        suMitarbeiterManager = new SuMitarbeiterManagerImpl();
    }

    @Override
    public SuMitarbeiter add(SuMitarbeiter entity) throws RemoteException,
Exception {
        return suMitarbeiterManager.add(entity);
    }

    @Override
    public SuMitarbeiter update(SuMitarbeiter entity) throws
RemoteException, Exception {
        return suMitarbeiterManager.update(entity);
    }

    @Override
    public void delete(SuMitarbeiter entity) throws RemoteException,
Exception {
        suMitarbeiterManager.delete(entity);
    }


    @Override
    public void deleteById(Integer id) throws Exception {
        suMitarbeiterManager.deleteById(id);
    }

    @Override
    public SuMitarbeiter findById(Integer id) throws RemoteException {
        return suMitarbeiterManager.findById(id);
    }

    @Override
    public List<SuMitarbeiter> findAll() throws RemoteException {
        return suMitarbeiterManager.findAll();
    }
}
```

MangelManagerService

```
package ch.hsluw.mangelmanager.webservice;

import java.util.List;

import javax.jws.WebMethod;
import javax.jws.WebService;
```

```
import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;

@WebService
public interface MangelManagerService {

    @WebMethod
    public abstract List<Projekt> getAllProjekt();
    @WebMethod
    public abstract List<Subunternehmen> getAllSubunternehmen();
    @WebMethod
    public abstract String getProjektproSubunternehmen(int subunternehmen);
    @WebMethod
    public abstract List<Mangel> getAllMangel();
    @WebMethod
    public abstract List<Meldung> getAllMeldung();
    @WebMethod
    public abstract Projekt getProjektById(int projektId);
    @WebMethod
    public abstract Subunternehmen getSubunternehmenById(int
subunternehmenId);
    @WebMethod
    public abstract Meldung getMeldungById(int meldungId);
    @WebMethod
    public abstract List<Person> getAllPerson();
    @WebMethod
    public abstract void updateSubunternehmen(Subunternehmen
subunternehmen);
    @WebMethod
    public abstract List<Projekt> getProjektByBezeichnung(String
bezeichnung);
    @WebMethod
    public abstract List<Plz> getAllPlz();
    @WebMethod
    public abstract List<Projekt> getProjektByBauherr(String bauherr);
    @WebMethod
    public abstract List<Projekt> getProjektByPlz(String plz);
    @WebMethod
    public abstract List<Projekt> getProjektByOrt(String ort);
    @WebMethod
    public abstract Plz getPlzById(int plzId);
    @WebMethod
    public abstract List<Projekt> getProjektByObjekttyp(String objekttyp);
    @WebMethod
    public abstract List<Projekt> getProjektByArbeitstyp(String
arbeitstyp);
```

```
    @WebMethod
    public abstract List<Projekt> getProjektByProjektstatus(String
projektstatus);
    @WebMethod
    public abstract void addAdresse(Adresse adresse);
    @WebMethod
    public abstract void addSubunternehmen(Subunternehmen
addSubunternehmen);
    @WebMethod
    public abstract List<Projekt> getAllSubunternehmenProjekt(
            Integer subunternehmen);
    @WebMethod
    public abstract List<SuMitarbeiter> getAllSubunternehmenMitarbeiter(
            Subunternehmen subunternehmen);
    @WebMethod
    public abstract List<Objekttyp> getAllObjekttyp();
    @WebMethod
    public abstract List<Arbeitstyp> getAllArbeitstyp();
    @WebMethod
    public abstract List<Mangel> getAllMangelProjekt(Integer projekt);
    @WebMethod
    public abstract Mangel getMangelById(Integer mangelId);
    @WebMethod
    public abstract void updateMangel(Mangel mangel);
    @WebMethod
    public abstract List<Mangelstatus> getAllMangelStatus();
    @WebMethod
    public abstract void addMangel(Mangel mangel);
    @WebMethod
    public abstract List<Meldungstyp> getAllMeldungstyp();
    @WebMethod
    public abstract void addMeldung(Meldung meldung);
    @WebMethod
    public abstract List<Meldung> getAllMeldungByMangel(Mangel mangel);
    @WebMethod
    public abstract List<Subunternehmen> getUnternehmenByProjekt(
            Integer projekt2);
    @WebMethod
    public abstract List<ProjektGuMitarbeiter> getBauleiterByProjekt(
            Projekt projekt2);
    @WebMethod
    public abstract Login getLoginByName(String name);
    @WebMethod
    public abstract Login getLoginById(Integer loginId);
    @WebMethod
    public abstract void updateProjekt(Projekt projekt2);
    @WebMethod
    public abstract void addSuMitarbeiterByProjekt(ProjektSuMitarbeiter
psum);
    @WebMethod
    public abstract List<Bauherr> getAllBauherr();
    @WebMethod
    public abstract void addProjekt(Projekt projekt2);
    @WebMethod
    public abstract List<GuMitarbeiter> getAllGuMitarbeiter();
    @WebMethod
    public abstract void addGuMitarbeiterByProjekt(
            ProjektGuMitarbeiter projektGuMitarbeiter);
    @WebMethod
    public abstract void updateProjektGuMitarbeiter(ProjektGuMitarbeiter
lastBauleiter);
    @WebMethod
    public abstract List<Projektstatus> getAllProjektstatus();
```

```
    @WebMethod
    public abstract void addGuMitarbeiter(GuMitarbeiter guMitarbeiter);
    @WebMethod
    public abstract void addSuMitarbeiter(SuMitarbeiter suMitarbeiter);
    @WebMethod
    public abstract void addBauherr(Bauherr bauherr);
    @WebMethod
    public abstract List<Rolle> getAllRolle();
    @WebMethod
    public abstract Person getPersonById(int personId);
    @WebMethod
    public abstract List<Projekt> getProjektbyPerson(Person person);
    @WebMethod
    public abstract void updatePerson(Person person);
    @WebMethod
    public abstract void updateMeldung(Meldung meldung2);
}
```

MangelManagerServiceImpl

```
package ch.hsluw.mangelmanager.webservice;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.List;
import java.util.Properties;

import javax.jws.WebService;

import ch.hsluw.mangelmanager.model.Adresse;
import ch.hsluw.mangelmanager.model.Arbeitstyp;
import ch.hsluw.mangelmanager.model.Bauherr;
import ch.hsluw.mangelmanager.model.GuMitarbeiter;
import ch.hsluw.mangelmanager.model.Login;
import ch.hsluw.mangelmanager.model.Mangel;
import ch.hsluw.mangelmanager.model.Mangelstatus;
import ch.hsluw.mangelmanager.model.Meldung;
import ch.hsluw.mangelmanager.model.Meldungstyp;
import ch.hsluw.mangelmanager.model.Objekttyp;
import ch.hsluw.mangelmanager.model.Person;
import ch.hsluw.mangelmanager.model.Plz;
import ch.hsluw.mangelmanager.model.Projekt;
import ch.hsluw.mangelmanager.model.ProjektGuMitarbeiter;
import ch.hsluw.mangelmanager.model.ProjektSuMitarbeiter;
import ch.hsluw.mangelmanager.model.Projektstatus;
import ch.hsluw.mangelmanager.model.Rolle;
import ch.hsluw.mangelmanager.model.SuMitarbeiter;
import ch.hsluw.mangelmanager.model.Subunternehmen;
import ch.hsluw.mangelmanager.rmi.adresse.AdresseRO;
import ch.hsluw.mangelmanager.rmi.arbeitstyp.ArbeitstypRO;
import ch.hsluw.mangelmanager.rmi.bauherr.BauherrRO;
import ch.hsluw.mangelmanager.rmi.gumitarbeiter.GuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.login.LoginRO;
import ch.hsluw.mangelmanager.rmi.mangel.MangelRO;
import ch.hsluw.mangelmanager.rmi.mangelstatus.MangelstatusRO;
import ch.hsluw.mangelmanager.rmi.meldung.MeldungRO;
import ch.hsluw.mangelmanager.rmi.meldungstyp.MeldungstypRO;
import ch.hsluw.mangelmanager.rmi.objekttyp.ObjekttypRO;
import ch.hsluw.mangelmanager.rmi.person.PersonRO;
import ch.hsluw.mangelmanager.rmi.plz.PlzRO;
import ch.hsluw.mangelmanager.rmi.projekt.ProjektRO;
```

```
import
ch.hsluw.mangelmanager.rmi.projektgumitarbeiter.ProjektGuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.projektstatus.ProjektstatusRO;
import
ch.hsluw.mangelmanager.rmi.projektsumitarbeiter.ProjektSuMitarbeiterRO;
import ch.hsluw.mangelmanager.rmi.rolle.RolleRO;
import ch.hsluw.mangelmanager.rmi.subunternehmen.SubunternehmenRO;
import ch.hsluw.mangelmanager.rmi.sumitarbeiter.SuMitarbeiterRO;

@WebService(endpointInterface =
"ch.hsluw.mangelmanager.webservice.MangelManagerService")
public class MangelManagerServiceImpl implements MangelManagerService{

    List<Person> person;
    List<Projekt> projekte;
    List<Projekt> suprojekte;
    List<Projektstatus> projektstatus;
    List<Subunternehmen> subunternehmen;

    List<Mangel> maengel;
    List<Meldung> meldung;
    List<Plz> plz;
    List<Objekttyp> objekttyp;
    List<Arbeitstyp> arbeitstyp;
    List<SuMitarbeiter> sumitarbeiter;
    List<ProjektGuMitarbeiter> bauleiter;
    List<Mangel> mangelOfProjekt;
    List<Meldung> meldungByMangel;
    List<Bauherr> bauherren;
    List<GuMitarbeiter> guMitarbeiter;

    List<Mangelstatus> mangelstatus;
    List<Meldungstyp> meldungstyp;
    String anzProjekte;
    Projekt projekt;
    Subunternehmen subunternehmennr;
    Meldung meldungnr;
    Plz plznr;
    Mangel mangelnr;
    Adresse addAdresse;
    Login login;
    Login loginnr;
    Person personnr;
    List<Rolle> rollen;

    PersonRO personRO;
    ProjektRO projektRO;
    SubunternehmenRO subunternehmenRO;
    MangelRO mangelRO;
    MeldungRO meldungRO;
    PlzRO plzRO;
    AdresseRO adresseRO;
    ObjekttypRO  objekttypRO;
    ArbeitstypRO arbeitstypRO;
    MangelstatusRO mangelstatusRO;
    MeldungstypRO meldungstypRO;
    ProjektGuMitarbeiterRO projektGuMitarbeiterRO;
    LoginRO loginRO;
    ProjektSuMitarbeiterRO projektSuMitarbeiterRO;
    BauherrRO bauherrRO;
    GuMitarbeiterRO guMitarbeiterRO;
    ProjektstatusRO projektstatusRO;
    SuMitarbeiterRO suMitarbeiterRO;
```

```
    RolleRO rolleRO;

    public MangelManagerServiceImpl() throws Exception {
        /*
         * Host-IP und RMI-Port definieren
         */
        String hostIp;
        int rmiPort;
        String url;

        // SecurityManager braucht nicht installiert zu werden, da Tomcat
einen
        // eigenen SecurityManager hat

        try {

            /* Properties laden */
            Properties props = new Properties();

            InputStream is =
ManglelManagerServiceImpl.class.getClassLoader()
                    .getResourceAsStream("ws.properties");

            props.load(is);

            hostIp = props.getProperty("rmi.host_ip");
            rmiPort = Integer.parseInt(props.getProperty("rmi.port"));
            url = "rmi://" + hostIp + ":" + rmiPort + "/";

            // URLs definieren

            String personROName = "personRO";
            String projektROName = "projektRO";
            String subunternehmenROName = "subunternehmenRO";
            String mangelROName = "mangelRO";
            String meldungROName ="meldungRO";
            String plzROName ="plzRO";
            String adresseROName = "adresseRO";
            String objekttypROName = "objekttypRO";
            String arbeitstyROName = "arbeitstypRO";
            String mangelstatusROName = "mangelstatusRO";
            String meldungstypROName = "meldungstypRO";
            String projektGuMitarbeiterROName = "projektGuMitarbeiterRO";
            String loginROName = "loginRO";
            String projektSuMitarbeiterROName = "projektSuMitarbeiterRO";
            String bauherrROName = "bauherrRO";
            String guMitarbeiterROName = "guMitarbeiterRO";
            String suMitarbeiterROName = "suMitarbeiterRO";
            String projektstatusROName = "projektstatusRO";
            String rolleROName = "rolleRO";

            this.personRO = (PersonRO) Naming.lookup(url + personROName);
            this.projektRO = (ProjektRO) Naming.lookup(url +
projektROName);
            this.mangelRO = (MangelRO) Naming.lookup(url + mangelROName);
            this.meldungRO = (MeldungRO) Naming.lookup(url +
meldungROName);
            this.subunternehmenRO = (SubunternehmenRO) Naming.lookup(url +
subunternehmenROName);
            this.plzRO = (PlzRO) Naming.lookup(url + plzROName);
            this.adresseRO = (AdresseRO) Naming.lookup(url +
adresseROName);
```

```
            this.objekttypRO = (ObjekttypRO) Naming.lookup(url +
objekttypROName);
            this.arbeitstypRO = (ArbeitstypRO) Naming.lookup(url +
arbeitstyROName);
            this.mangelstatusRO = (MangelstatusRO) Naming.lookup(url +
mangelstatusROName);
            this.meldungstypRO = (MeldungstypRO) Naming.lookup(url +
meldungstypROName);
            this.projektGuMitarbeiterRO = (ProjektGuMitarbeiterRO)
Naming.lookup(url + projektGuMitarbeiterROName);
            this.loginRO = (LoginRO) Naming.lookup(url+ loginROName);
            this.projektSuMitarbeiterRO = (ProjektSuMitarbeiterRO)
Naming.lookup(url + projektSuMitarbeiterROName);
            this.bauherrRO = (BauherrRO) Naming.lookup(url +
bauherrROName);
            this.guMitarbeiterRO = (GuMitarbeiterRO) Naming.lookup(url +
guMitarbeiterROName);
            this.projektstatusRO = (ProjektstatusRO) Naming.lookup(url +
projektstatusROName);
            this.suMitarbeiterRO = (SuMitarbeiterRO) Naming.lookup(url +
suMitarbeiterROName);
            this.rolleRO = (RolleRO) Naming.lookup(url + rolleROName);

        } catch (MalformedURLException | RemoteException |
NotBoundException e) {
            throw e;
        }
    }


    public List<Projekt> getAllProjekt() {
        // TODO Auto-generated method stub
        try {
            projekte = projektRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;

    }

    public List<Subunternehmen> getAllSubunternehmen() {

        try {
            subunternehmen = subunternehmenRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return subunternehmen;
    }

    public String getProjektproSubunternehmen(int subunternehmen){
        try {
            anzProjekte = subunternehmenRO.findAllProjekte(subunternehmen);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return anzProjekte;

    }
```

```
public List<Mangel> getAllMangel() {

    // TODO Auto-generated method stub
    try {
        maengel = mangelRO.findAll();
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return maengel;

}


public List<Meldung> getAllMeldung(){
    try {
        meldung = meldungRO.findAll();
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return meldung;
}

public Projekt getProjektById(int projektId) {
    // TODO Auto-generated method stub
    try {
        projekt = projektRO.findById(projektId);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return projekt;
}

public Subunternehmen getSubunternehmenById(int subunternehmenId) {
    try {
        subunternehmennr = subunternehmenRO.findById(subunternehmenId);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return subunternehmennr;
}

public Meldung getMeldungById(int meldungId) {
    try {
        meldungnr = meldungRO.findById(meldungId);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return meldungnr;
}

public List<Person> getAllPerson() {
    // TODO Auto-generated method stub
    try {
        person = personRO.findAll();
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
            return person;
    }


    public void updateSubunternehmen(Subunternehmen subunternehmen) {
            try {
                subunternehmenRO.update(subunternehmen);
            } catch (RemoteException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }


    public List<Projekt> getProjektByBezeichnung(String bezeichnung) {
            // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByBezeichnung(bezeichnung);
            for (Projekt projekt : projekte) {
                System.out.println(projekt.getBeschreibung());
            }
        }
        catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Plz> getAllPlz() {
        try {
            plz = plzRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return plz;

    }

    public List<Projekt> getProjektByBauherr(String bauherr) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByBauherr(bauherr);
            for (Projekt projekt : projekte) {

System.out.println(projekt.getFkBauherr().get(0).getNachname());
            }
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByPlz(String plz) {
        // TODO Auto-generated method stub
```

```
        try {
            projekte =  projektRO.findByPlz(plz);
            for (Projekt projekt : projekte) {

System.out.println(projekt.getFkAdresse().getPlz().getPlz());
            }
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByOrt(String ort) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByOrt(ort);
            for (Projekt projekt : projekte) {

System.out.println(projekt.getFkAdresse().getPlz().getOrt());
            }
        }
        catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public Plz getPlzById(int plzId) {
        try {
            plznr = plzRO.findById(plzId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return plznr;
    }

    public List<Projekt> getProjektByObjekttyp(String objekttyp) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByObjekttyp(objekttyp);
            for (Projekt projekt : projekte) {

System.out.println(projekt.getFkObjekttyp().getBezeichnung());
            }
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByArbeitstyp(String arbeitstyp) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByArbeitstyp(arbeitstyp);
            for (Projekt projekt : projekte) {

System.out.println(projekt.getFkArbeitstyp().getBezeichnung());;
            }
        } catch (RemoteException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }

    public List<Projekt> getProjektByProjektstatus(String projektstatus) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findByProjektstatus(projektstatus);
            for (Projekt projekt : projekte) {

System.out.println(projekt.getFkProjektstatus().getBezeichnung());
            }
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return projekte;
    }


    public void addAdresse(Adresse adresse) {
        try {
            adresseRO.add(adresse);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void addSubunternehmen(Subunternehmen addSubunternehmen) {
        try {
            subunternehmenRO.add(addSubunternehmen);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public List<Projekt> getAllSubunternehmenProjekt(Integer
subunternehmen) {
        try {
            suprojekte =
projektRO.findAllSubunternehmenProjekt(subunternehmen);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return suprojekte;
    }

    public List<SuMitarbeiter>
getAllSubunternehmenMitarbeiter(Subunternehmen subunternehmen) {
        try {
            sumitarbeiter =
subunternehmenRO.findAllSubunternehmenMitarbeiter(subunternehmen);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return sumitarbeiter;
```

```
    }

    public List<Objekttyp> getAllObjekttyp() {
        // TODO Auto-generated method stub
        try {
            objekttyp = objekttypRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return objekttyp;
    }

    public List<Arbeitstyp> getAllArbeitstyp() {
        // TODO Auto-generated method stub
        try {
            arbeitstyp = arbeitstypRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return arbeitstyp;
    }
    public List<Mangel> getAllMangelProjekt(Integer projekt) {
        try {
            mangelOfProjekt = mangelRO.findAllMangelProjekt(projekt);
//          System.out.println(mangelOfProjekt.get(0).getBezeichnung());
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return mangelOfProjekt;
    }

    public Mangel getMangelById(Integer mangelId) {
        try {
            mangelnr = mangelRO.findById(mangelId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return mangelnr;
    }

    public void updateMangel(Mangel mangel) {
        // TODO Auto-generated method stub
        try {
            mangelRO.update(mangel);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Mangelstatus> getAllMangelStatus() {
        try {
            mangelstatus = mangelstatusRO.findAllMangelStatus();
//          System.out.println(mangelOfProjekt.get(0).getBezeichnung());
        } catch (Exception e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
            return mangelstatus;
    }

    public void addMangel(Mangel mangel) {
        // TODO Auto-generated method stub
        try {
            mangelRO.add(mangel);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Meldungstyp> getAllMeldungstyp() {
        // TODO Auto-generated method stub
        try {
            meldungstyp = meldungstypRO.findAll();
//          System.out.println(mangelOfProjekt.get(0).getBezeichnung());
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return meldungstyp;
    }

    public void addMeldung(Meldung meldung) {
        // TODO Auto-generated method stub
        try {
            meldungRO.add(meldung);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Meldung> getAllMeldungByMangel(Mangel mangel) {
        // TODO Auto-generated method stub
        try {
            meldungByMangel = meldungRO.findAllMeldungByMangel(mangel);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return meldungByMangel;
    }

    public List<Subunternehmen> getUnternehmenByProjekt(Integer projekt2) {
        // TODO Auto-generated method stub
        try {
            subunternehmen =
subunternehmenRO.findAllSubunternehmenByProjekt(projekt2);
        } catch (Exception e) {
            // TODO Auto-generated catch block<
            e.printStackTrace();
        }
            return subunternehmen;
    }

    public List<ProjektGuMitarbeiter> getBauleiterByProjekt(Projekt
projekt2) {
        // TODO Auto-generated method stub
```

```
        try {
            bauleiter =
projektGuMitarbeiterRO.findAllBauleiterByProjekt(projekt2);
        } catch (Exception e) {
            e.printStackTrace();
        }
            return bauleiter;
    }

    public Login getLoginByName(String name) {
        // TODO Auto-generated method stub
        try {
            login = loginRO.findByName(name);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return login;
    }

    public Login getLoginById(Integer loginId) {
        // TODO Auto-generated method stub
        try {
            loginnr = loginRO.findById(loginId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return loginnr;
    }

    public void updateProjekt(Projekt projekt2) {
        // TODO Auto-generated method stub
        try {
            projektRO.update(projekt2);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void addSuMitarbeiterByProjekt(ProjektSuMitarbeiter psum) {
        // TODO Auto-generated method stub
        try {
            projektSuMitarbeiterRO.add(psum);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Bauherr> getAllBauherr() {
        // TODO Auto-generated method stub
        try {
            bauherren = bauherrRO.findAll();
//          System.out.println(mangelOfProjekt.get(0).getBezeichnung());
        } catch (Exception e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return bauherren;
    }

    public void addProjekt(Projekt projekt2) {
        // TODO Auto-generated method stub
        try {
            projektRO.add(projekt2);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<GuMitarbeiter> getAllGuMitarbeiter() {
        // TODO Auto-generated method stub
        try {
            guMitarbeiter = guMitarbeiterRO.findAll();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
            return guMitarbeiter;
    }

    public void addGuMitarbeiterByProjekt(
            ProjektGuMitarbeiter projektGuMitarbeiter) {
        // TODO Auto-generated method stub
        try {
            projektGuMitarbeiterRO.add(projektGuMitarbeiter);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public void updateProjektGuMitarbeiter(ProjektGuMitarbeiter
lastBauleiter) {
        // TODO Auto-generated method stub
        try {
            projektGuMitarbeiterRO.update(lastBauleiter);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Projektstatus> getAllProjektstatus() {
        // TODO Auto-generated method stub
        try {
            projektstatus = projektstatusRO.findAll();
```

```
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return projektstatus;
    }

    public void addGuMitarbeiter(GuMitarbeiter guMitarbeiter) {
        // TODO Auto-generated method stub
        try {
            guMitarbeiterRO.add(guMitarbeiter);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void addSuMitarbeiter(SuMitarbeiter suMitarbeiter) {
        // TODO Auto-generated method stub
        try {
            suMitarbeiterRO.add(suMitarbeiter);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void addBauherr(Bauherr bauherr) {
        // TODO Auto-generated method stub
        try {
            bauherrRO.add(bauherr);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public List<Rolle> getAllRolle() {
        try {
            rollen = rolleRO.findAll();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return rollen;
    }

    public Person getPersonById(int personId) {
        // TODO Auto-generated method stub
        try {
            personnr = personRO.findById(personId);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return personnr;
    }

    public List<Projekt> getProjektbyPerson(Person person) {
        // TODO Auto-generated method stub
        try {
            projekte =  projektRO.findProjektbyPerson(person);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
        return projekte;
    }

    public void updatePerson(Person person) {
        // TODO Auto-generated method stub
        try {
            personRO.update(person);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void updateMeldung(Meldung meldung2) {
        // TODO Auto-generated method stub
        try {
            meldungRO.update(meldung2);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }


}
```