

Chiappelloni Nicolas
Projet d'analyse et de conception
3 ème année
Professeur: Mr. Ninforge-ory
ECI

20 octobre 2019





TOLL MANAGER

Projet d'analyse et de conception

Table of contents

CONTEXT	4
TOOLKIT.....	6
PLANNING.....	7
AGILE MANIFEST	0
APPLICATION ARCHITECTURE.....	1
Business Layer	1
Model	1
Infrastructure.....	2
Application.....	2
DOMAIN DRIVEN DEVELOPMENT	3
Value Object	3
No anemic object	4
Null Object Pattern	5
Factory method	5
Domain separation.....	7
TEST DRIVEN DEVELOPMENT	8
BEHAVIOR DRIVEN DEVELOPMENT	10
Example of feature:.....	10
COMMUNICATE WITH GRAPHIC	12
Command:	12
example:.....	12
Query	15
example:.....	15
UML	18
USE CASES.....	18
Manage team	18
Manage Employee.....	18
CLASS DIAGRAM	20
Planning diagram	20
Search diagram	21
THESAURUS DATABASE	23
MCD.....	0
MLD.....	1
Addresses.....	2
Country.....	2
Cities	2
Address.....	2
Contacts.....	3
Phone	3
Email	3

Teams	4
Team	4
Planning	6
planning	6
Groups authorities.....	7
Groups.....	7
Rule	7
Employees and user.....	8
Employee	8
USER DOCUMENTATION	10
BIG PICTURES	11
AUTHORIZATION MANAGEMENT	13
 Team leader.....	13
Roles	13
Limitation	13
 Manager	13
Roles	13
Limitation	14
Administrator.....	14
APPLICATION LAUNCH.....	15
First launch	15
Authentication	16
EMPLOYEE MANAGEMENT.....	18
Create an employee	18
Create a team leader or a manager.....	19
Assign team leader to a team	19
Edit an employee.....	20
Delete an employee.....	21
Search an employee.....	22
TEAM MANAGEMENT	23
Create a root team	23
Create a sub team.....	24
Remove a main team	24
Remove a sub team	25
Append employee to team	26
Remove employee to team.....	26
PLANNING MANAGEMENT	28
Append a day planning	28
Remove a planning.....	30
Navigate on week.....	31
Show information.....	31

Context

One toll company for motorways needs an application that can manage planning for employee, each employee belongs to a team that be managed by a team leader. The team leader is responsible to manage his teams and his members they've attributed by the human resources. The application is used mostly for team leader to manage his weeks planning, and by the human resources that give to team leader main teams and employees.

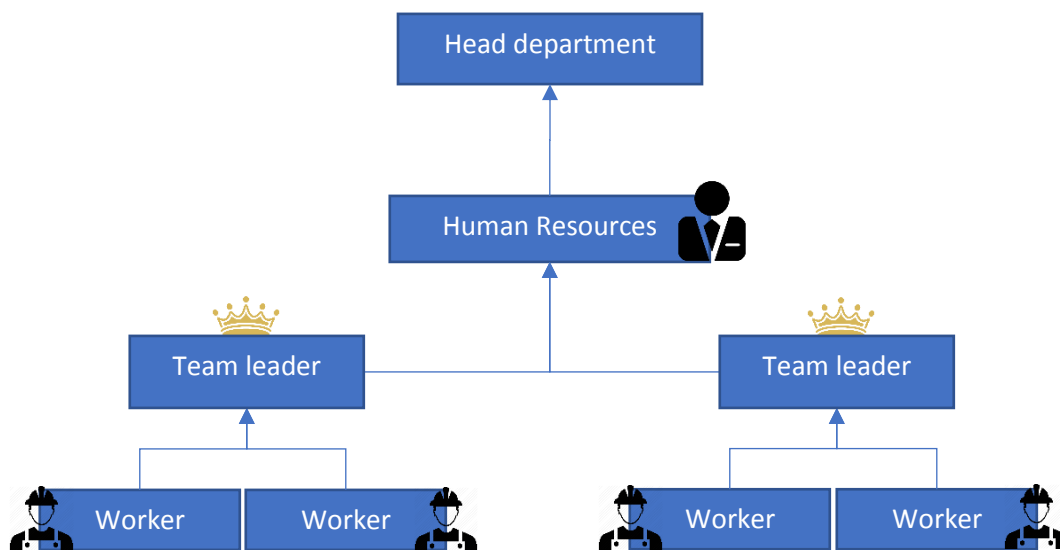


Figure 1 organizational chart

Application usage

- Worker
 - No access to the application
- Team leader
 - responsible to one or many teams
 - create, edit or remove sub teams
 - arrange employee to his sub team
 - create, edit or remove planning for each employee
 - cannot delete or edit employee
 - cannot remove employee from a “main team”
- Human ressources

- create, edit or remove employee
 - assign employee to team
 - Create, edit or remove team
 - create, edit or remove planning
- Head department
 - Head department have an access to do everything.

Toolkit

Language: Java 11

IDE : IntelliJ IDEA community version

Framework used:

- OpenjavaFX 13 (graphic component)
- Junit 4.12 (test framework)
- TestFx (graphic test framework)
- Hamcrest 2.1 (expression match , used for test)
- Mockito 3.0 (mock object for test)
- Cucumber 4.7 (behavior driven developpement)
- PostgresSqlDriver 42.2

Database : postgres

DB name : tollManagerV2

Planning

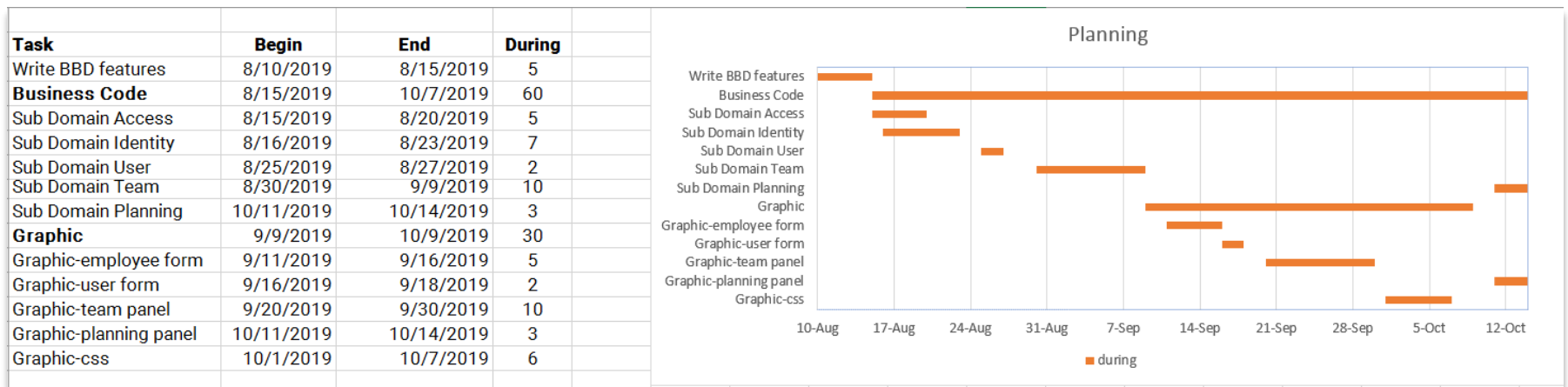


Figure 2dev planning

Agile Manifest

Application Architecture

The application is sliced into 2 big parts, the graphic part and the business part linked by the *applicationLifeTime* interface, in this interface we put every use case which the graphic application uses to deal with the business code (see the **** CQS borrow model ****).

Business Layer

In the business layer we have 3 main sections.

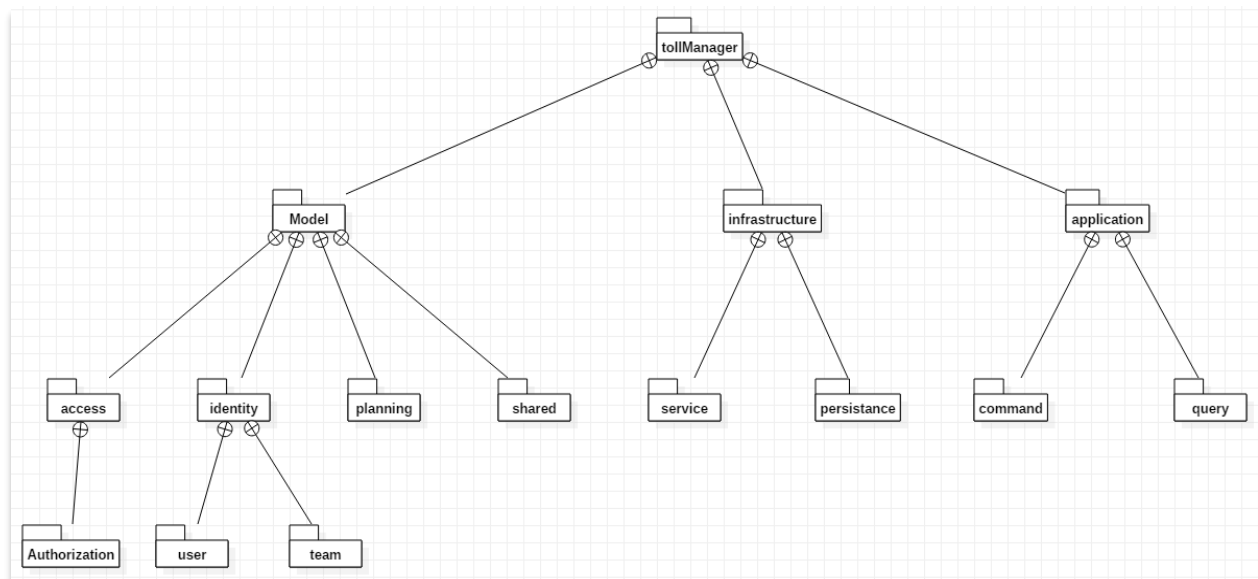


Figure 3business layer

Model

In model section we have the Domain of the application where each sub domain is packaged.

Each sub domain must be agnostic from other sub domain for example the sub domain “access” doesn’t care about the sub domain “identity” or “planning”, But if for some measured reason something needed to be shared between two subdomains the package *shared* exists, but the

reason to do that should be precise (example an interface observable or class responsible to generate id, ...).

(don't confuse model from MVC and here from DDD it's same purpose but in DDD the model has some responsibility and rules than MVC doesn't precise).

Infrastructure

Here we put everything that depends of a specific architecture, like the repository(postgres, mysql, ...) or service like cryptography(md5,sha-1,...)

Application

Here we put what is concerned by the launching of the application like *Command* or *Query* for CQS or The ApplicationLifeTime implementation.

The application layer is like the cement that link all part of the model

Domain Driven Development

Some of strategies are borrow from the domain driven design, we don't respect every concept (cause too complex) but some interesting concepts are used.

Value Object

When you need something like the name of a person don't use a string to represent the name but use a value object, A small object **immutable** that represents a simple entity whose equality is not based on identity.

```
public class Niss {
    private final String niss;
    public static final String REGEX="^\\d{2}\\.\\.\\d{2}\\.\\.\\d{2}\\.\\.\\d{3}\\.\\.\\d{2}$";

    private Niss(String niss) { this.niss = niss; }

    public static Niss of(String niss) {
        Objects.requireNonNull(niss, message: "The niss cannot be null.");
        if(!niss.matches(REGEX))
            throw new IllegalArgumentException("the niss format must be xx.xx.xx-xxx.xx");
        return new Niss(niss);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Niss)) return false;
        Niss other = (Niss) o;
        return Objects.equals(niss, other.niss);
    }

    @Override
    public int hashCode() { return Objects.hash(niss); }

    @Override
    public String toString() {
        return "Niss{" +
            "niss='" + niss + '\'' +
            '}';
    }

    public String value() { return niss; }
}
```

Figure 4: example of value object

Setter is not allowed! if you want to edit a value object then you need to create a new value object! (avoid side effect).

No anemic object

An object anemic is an object does nothing and let other object like service to do for its, it's not an OOP practice, an object is created to do something!

```
public class Person {  
  
    private String name;  
    private String forename;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getForename() {  
        return forename;  
    }  
  
    public void setForename(String forename) {  
        this.forename = forename;  
    }  
}
```

Figure 5 anemic model

```
public class Person {  
  
    private String name;  
    private String forename;  
  
    public String getName() { return name; }  
    public String getForename() { return forename; }  
    public String getNameFirstLetterCapital(){...}  
    public String getForenameFirstLetterCapital(){...}  
    public String getFullNameToString() { return getNameFirstLetterCapital()+" "+getForenameFirstLetterCapital(); }
```

Figure 6 no anemic model

As you can see on the picture 1, class does nothing just getter and setter against the model on figure 2 non anemic model where the object does something.

Null Object Pattern

Avoid null instead use the Null object pattern.

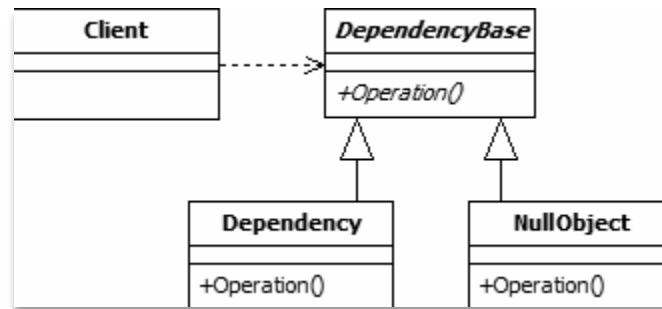


Figure 7 null object pattern

Factory method

In the business code, constructor is not allowed, we need to use the factory method pattern and for complex object, the Builder pattern

```
public class Group {
    private GroupName name;
    private LinkedHashSet<Role> roles;
    private LinkedHashSet<GroupMember> members;

    private Group(GroupName name, LinkedHashSet<Role> roles, LinkedHashSet<GroupMember> members) {
        this.name = name;
        this.roles = roles;
        this.members = members;
    }

    public static Group of(GroupName name, LinkedHashSet<Role> roles, LinkedHashSet<GroupMember> members) {
        Objects.requireNonNull(name, message: "The group name is required.");
        Objects.requireNonNull(roles, message: "The role list is required.");
        Objects.requireNonNull(members, message: "The member list required.");

        return new Group(name, roles, members);
    }
}
```

Figure 8 factory method instead constructor

```

public class GroupBuilder {
    private GroupName name;
    protected LinkedHashSet<Role> roles;
    protected LinkedHashSet<GroupMember> members;

    private GroupBuilder() {
        roles=new LinkedHashSet<>();
        members=new LinkedHashSet<>();
    }

    public static GroupBuilder of() { return new GroupBuilder(); }

    public GroupBuilder setName(GroupName name) {
        Objects.requireNonNull(name);
        this.name = name;
        return this;
    }

    public GroupBuilder setName(String name) {
        Objects.requireNonNull(name);
        this.name = GroupName.of(name);
        return this;
    }

    public GroupBuilder setRoles(LinkedHashSet<Role> roles) {
        Objects.requireNonNull(roles);
        this.roles = roles;
        return this;
    }

    public GroupBuilder addRole(Role role) {
        Objects.requireNonNull(role);
        roles.add(role);
        return this;
    }
}

```

Figure 9 GroupBuilder

```

Group aGroup=GroupBuilder.of()
    .setName("A group name")
    .addRole(Role.create_team())
    .addRole(Role.remove_member(GroupName.wildCard()))
    .setMembers(new LinkedHashSet<>())
    .create();

```

Figure 10use a builder

Domain separation

A domain it's a group of classes that can work together without other object extern to the domain, for example the planning domain don't care about the access, but when the ApplicationLifeTime will run, then the application can deal with access and planning together. See sub domain like a brick and the Application layer like the wall.

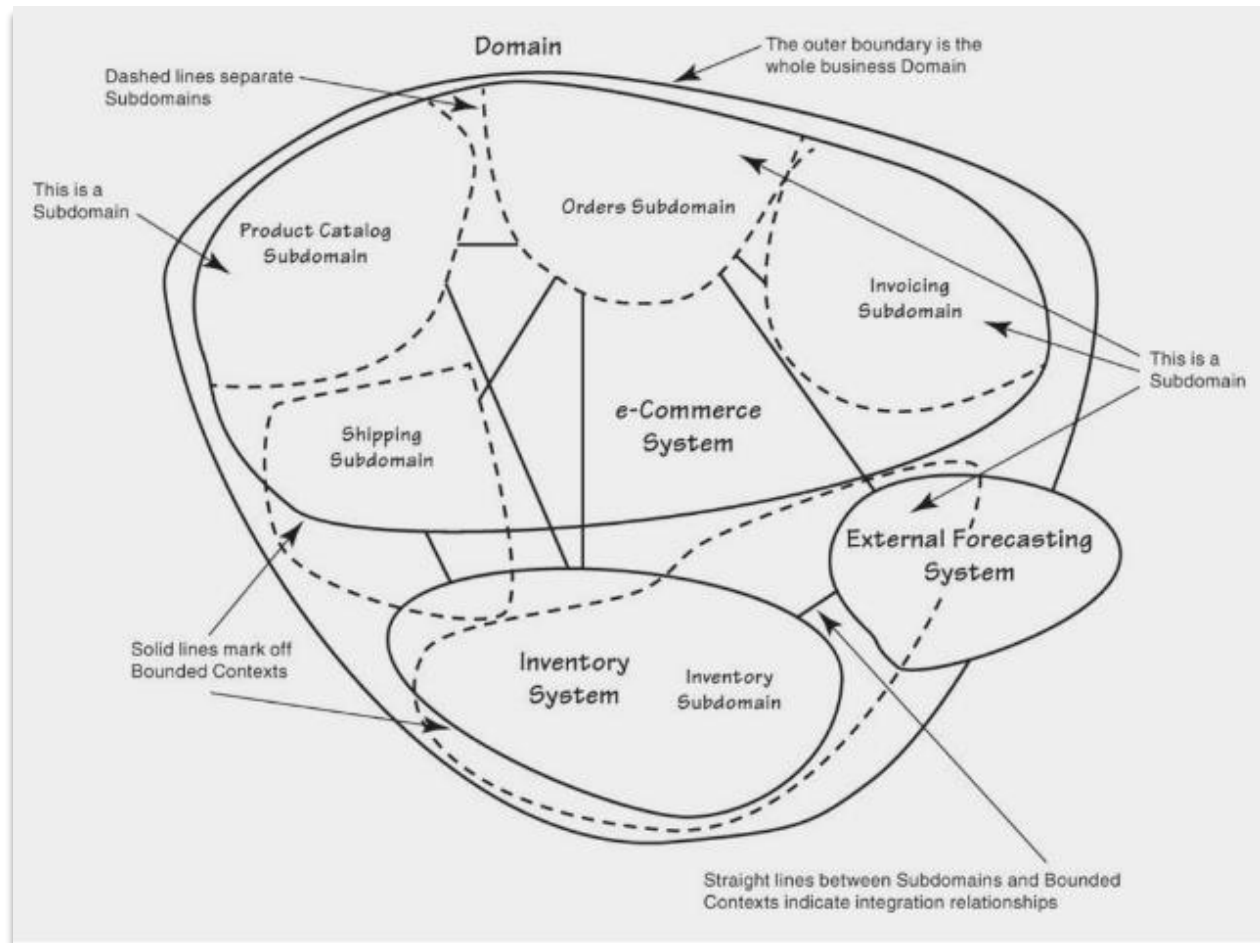


Figure 11 Domain example

Test Driven Development

Perhaps the most important part of this Manifest, No one nano octet of code should be writing without a test before into the business layer ! The 100% code coverage is not required (mean don't test toString and getter method) but It's very important to assure one piece of code appended or edited don't make a side effect on de business code !

As a reminder, when you want to write code, write the test before, launch the test to failed it, make code to pass test to green and refactor as far as it's possible.

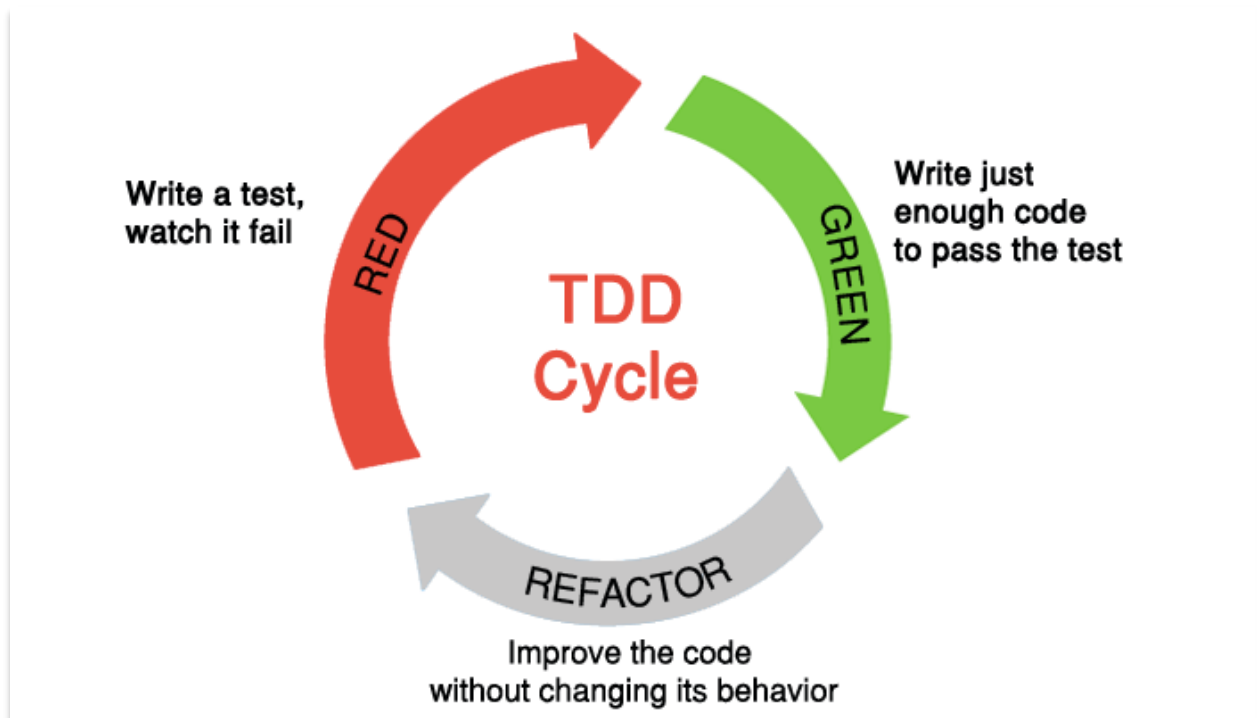


Figure 12tdd cycle

```

public class AuthenticationServiceTest {
    private UserRepository repository;
    private IAuthenticationService service;
    private IPasswordEncryptionService encryptionService;

    @Before
    public void setUp() {
        repository=new InMemoryUserRepository();
        encryptionService=new MD5PasswordEncryptionService();
        service=new AuthenticationService(repository,encryptionService);
    }

    @Test
    public void signIn_shouldReturnTheUser_whenLoginAndPasswordMatch() {...}
    @Test(expected = AuthenticationException.class)
    public void signIn_shouldThrowAuthenticationException_whenPasswordNotMatch() {...}
    @Test(expected = AuthenticationException.class)
    public void signIn_shouldThrowAuthenticationException_whenLoginNotMatch() {...}
}

```

Figure 13: TDD class example

As you can see, I began by test the simple thing, return user match login and password, when it's pass, I create a new test where I want to throw exception, I pass the test and assure my first test stay green, ... If I need to make one modification to my code a launch my test that assure I haven't a side effect...

Behavior Driven Development

Another big part of the manifest it's the BDD, when you want to append a feature to the Business code Then it's very important to write a Cucumber file that explains the feature! If you cannot write a feature maybe that mean you don't need to it...

Example of feature:

```
Feature: user management

Background:
  Given I have these users
    | employeeId | login | password |
    | 01         | admin | Secret123@ |

Scenario Outline: change password
  Given I'm connected like "<user>"
  When I change my password by "<newPassword>" "<newPasswordConfirmation>"
  Then my password should by "<password expected>"
  And the error message should be "<errorMessage>"
  And I'm connected with "<user>" "<password expected>"

Examples: success
  | user | newPassword | newPasswordConfirmation | password expected | errorMessage |
  | admin | Admin123@ | Admin123@ | Admin123@ | |

Examples: failure
  | user | newPassword | newPasswordConfirmation | password expected | errorMessage |
  | admin | notSecure | notSecure | Secret123@ | The password must contain 5 to 55 characters |
  | admin | Admin123@ | AnotherPass456@ | Secret123@ | Password does not match. |
```

Figure 14:BDD example

```
@Given("I'm connected like {login}")
public void i_m_connected_like(Login login) {
    try{
        helper.setConnectedUser(helper.userRepository().findByLogin(login));
    }catch (AuthenticationException e)
    {
        helper.setErrorMessage(e.getMessage());
    }
}
```

Figure 15first step in java

Feature: user management

▼ Scenario Outline: change password

Given I'm connected like "<user>"

When I change my password by "<newPassword>" "<newPasswordConfirmation>"

Then my password should be "<password expected>"

And the error message should be "<errorMessage>"

And I'm connected with "<user>" "<password expected>"

▼ Examples: success

user	newPassword	newPasswordConfirmation	password expected	errorMessage
admin	Admin123@	Admin123@	Admin123@	

► Background:

► Scenario Outline: change password

► Background:

► Scenario Outline: change password

► Background:

▼ Scenario Outline: change password

Given I'm connected like "admin"

When I change my password by "Admin123@" "AnotherPass456@"

Then my password should be "Secret123@"

And the error message should be "Passwofrd does not match."

```
org.junit.ComparisonFailure: expected:<Passwo[f]rd does not match.> but was:<Passwo[]rd does not match.>
    at org.junit.Assert.assertEquals(Assert.java:115)
    at org.junit.Assert.assertEquals(Assert.java:144)
    at features.access.AccessSteps.the_error_message_should_be(AccessSteps.java:21)
    at *.the error message should be "Passwofrd does not match."(file:/C:/Users/userwindop/Downloads/tollManagerGraphic/src
```

And I'm connected with "admin" "Secret123@"

Figure 16BDD report with one error, step not finish

▼ Feature: user management

▼ Scenario Outline: change password

Given I'm connected like "<user>"

When I change my password by "<newPassword>" "<newPasswordConfirmation>"

Then my password should be "<password expected>"

And the error message should be "<errorMessage>"

And I'm connected with "<user>" "<password expected>"

▼ Examples: success

user	newPassword	newPasswordConfirmation	password expected	errorMessage
admin	Admin123@	Admin123@	Admin123@	

▼ Background:

Given I have these users

employeeId	login	password
01	admin	Secret123@

▼ Scenario Outline: change password

Given I'm connected like "admin"

When I change my password by "Admin123@" "Admin123@"

Then my password should be "Admin123@"

And the error message should be ""

And I'm connected with "admin" "Admin123@"

▼ Examples: failure

user	newPassword	newPasswordConfirmation	password expected	errorMessage
admin	notSecure	notSecure	Secret123@	The password must contain 5 to 55 characters , one or more
admin	Admin123@	AnotherPass456@	Secret123@	Password does not match.

► Background:

► Scenario Outline: change password

► Background:

► Scenario Outline: change password

Figure 17BDD report, step finish

Communicate with graphic

To communicate with the graphic (means out of the business layer), we use a technic borrow to the *CQS*, the Command and Query.

Command:

A command is an action like 'create an employee', 'change my hair color', ...

```
public class CreateEmployeeCommand {
    private Person person;
    private Team team;

    public CreateEmployeeCommand(Person person, Team team) {
        this.person = person;
        this.team = team;
    }

    public Person getPerson() { return person; }

    public void setPerson(Person person) { this.person = person; }

    public Team getTeam() { return team; }

    public void setTeam(Team team) { this.team = team; }
}
```

Figure 18Command example

example:

A simple feature, when I click on the button the data is incremented.

```
Feature: increment database when click on button

  Scenario: click on button
    Given I have a the 'increment button' on my window
    And My data on database equals '1'
    When I click on the button 'increment button'
    Then My data on database equals '2'
```

Figure 19: feature example command

To make an action into the database I create a class where the end of the name is *Command* and the class is located on `tollManager.application.command` package.

```
public class IncrementDatabaseCommand {
    private int coefficient;

    public IncrementDatabaseCommand(int coefficient) {
        this.coefficient = coefficient;
    }

    public int getCoefficient() {
        return coefficient;
    }

    public void setCoefficient(int coefficient) {
        this.coefficient = coefficient;
    }
}
```

Figure 20 IncrementDataBaseCommand

After the class creation, I create the function *execute* into the *ApplicationLifeTime* that take as argument the *IncrementDataBaseCommand*

```

/**
 * @param command contain the coefficient by increment the data on database
 * @see IncrementDatabaseCommand
 */
void execute(IncrementDatabaseCommand command);

```

Figure 21 Execute function on ApplicationLifeTime

and in my implementation of ApplicationLifeTime, I call the concerned business.

```

@Override
public void execute(IncrementDatabaseCommand command) {
    try {
        db.setAutoCommit(false);
        //Call service 1
        //Call service 2
        //do something ...
        db.connection().commit();
    } catch (Exception e) {
        showError(e.getMessage());
        db.rollback();
    }
    finally {
        db.setAutoCommit(true);
    }
}

```

Figure 22 Execute implementation

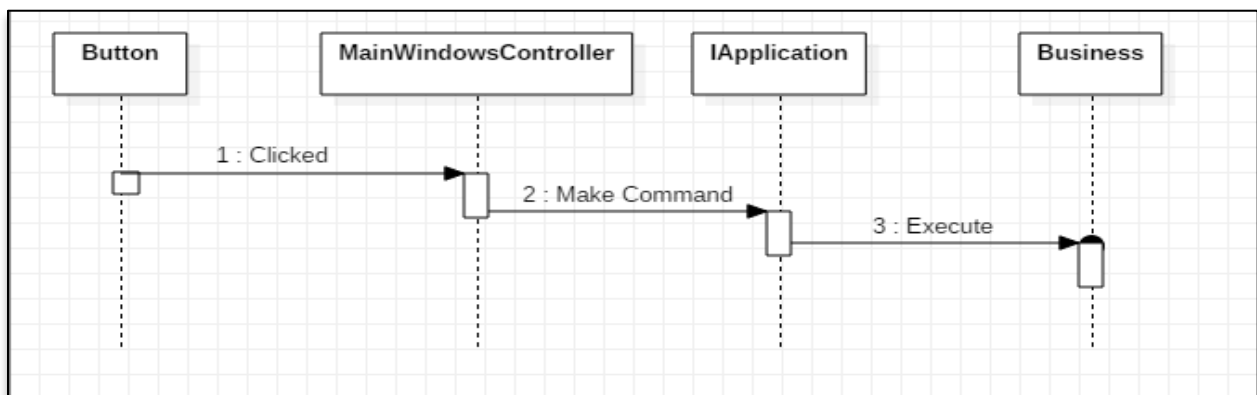


Figure 23: Command sequence

Query

A Query is an ask on the database or on the business layer, it does nothing , the business should reply something.

example:

```
Feature: ask database value when click on button

  Scenario: click on button
    Given I have a the 'ask button' on my window
    And My data on database equals '1'
    When I click on the button 'ask button'
    Then I see in my value label '1'
```

Figure 24query feature

To make a query to the database I create a class where the end of the name is *Query* and the class is located on tollManager.application.Query package.

```
public class DatabaseValueQuery {
    private Integer value;

    public DatabaseValueQuery() {
        this.value = null;
    }

    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }
}
```

Figure 25DatabaseValueQuery

After the class creation, I create the function *query* into the *ApplicationLifeTime* that take as argument the *DatabaseValueQuery*

```
/**
 * @param query contains the value in integer to returned
 * @see DatabaseValueQuery
 */
void query(DatabaseValueQuery query);
```

Figure 26query in ApplicationLifeTime

and in my implementation of *ApplicationLifeTime*, I call the concerned business, and I put the value in the query in argument.

```
@Override
public void query(DatabaseValueQuery query) {
    try{
        // int valueFromDatabase = database.getValue();
        // query.setValue(valueFromDatabase);
    }catch (Exception e){
        // query.setValue(-1);
    }
}
```

Figure 27Query implementation

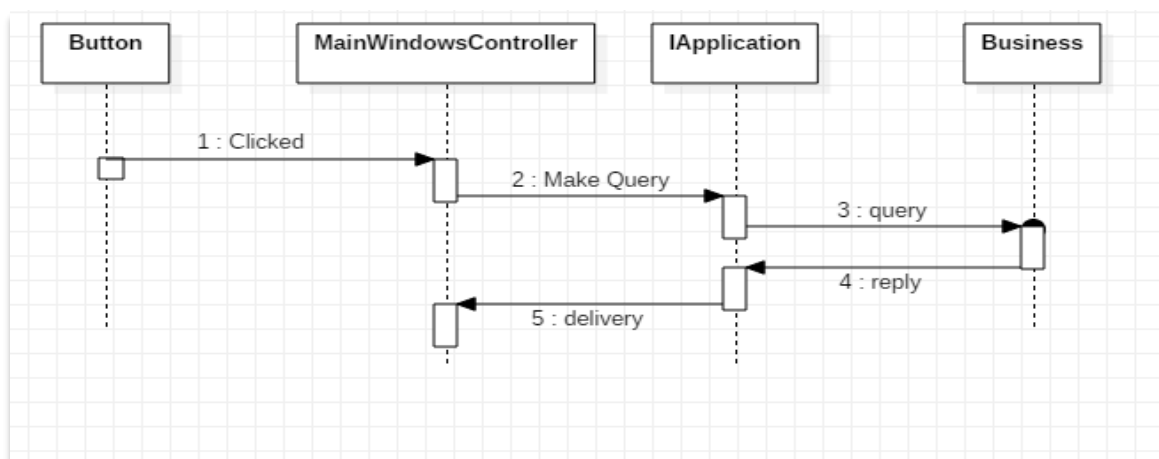


Figure 28: query execution

The advantage to use Command and Query it's a common way to deal between graphic and business without dependency from graphic or business!

Also, a big advantage It's Command and Query are incremental, so we can build a complex Query & Command during their journey.

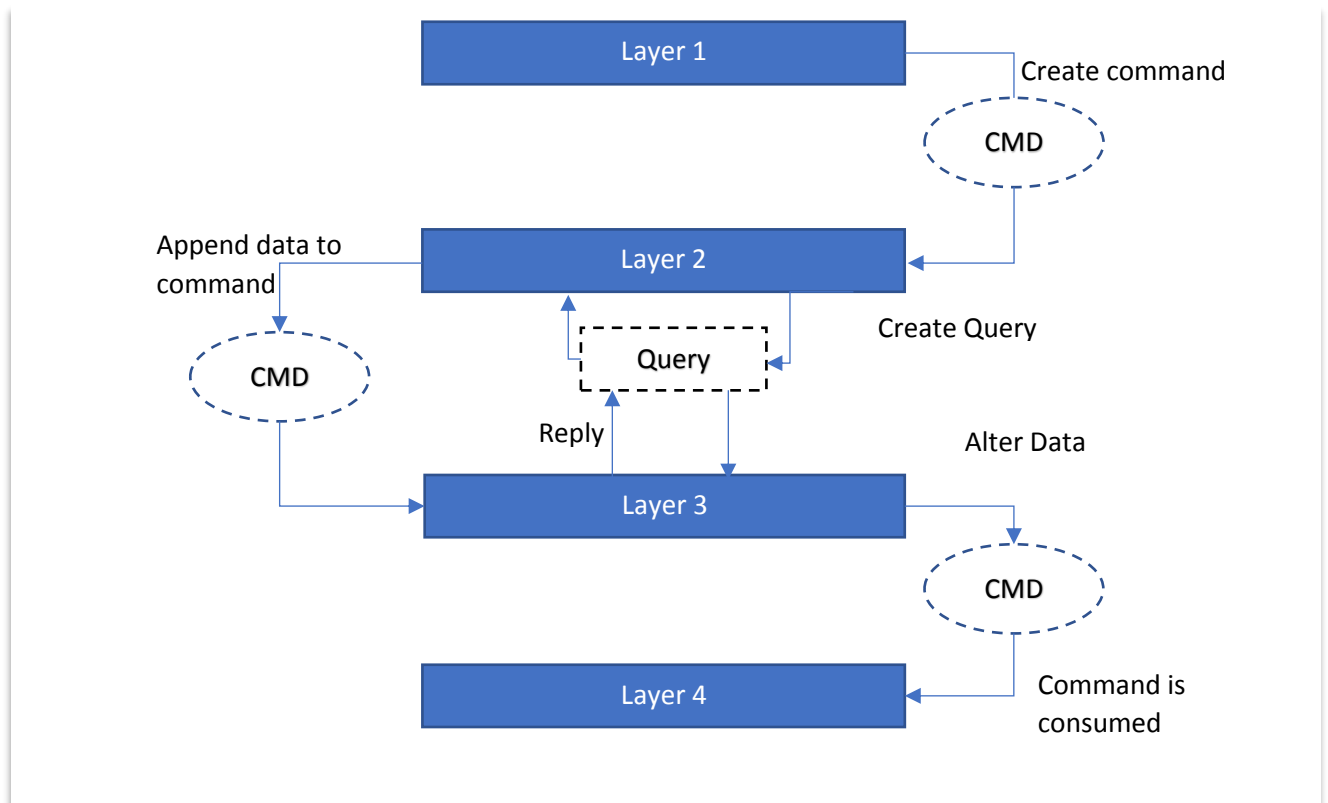


Figure 29 Example of complex path of a command

UML

Use Cases

Manage team

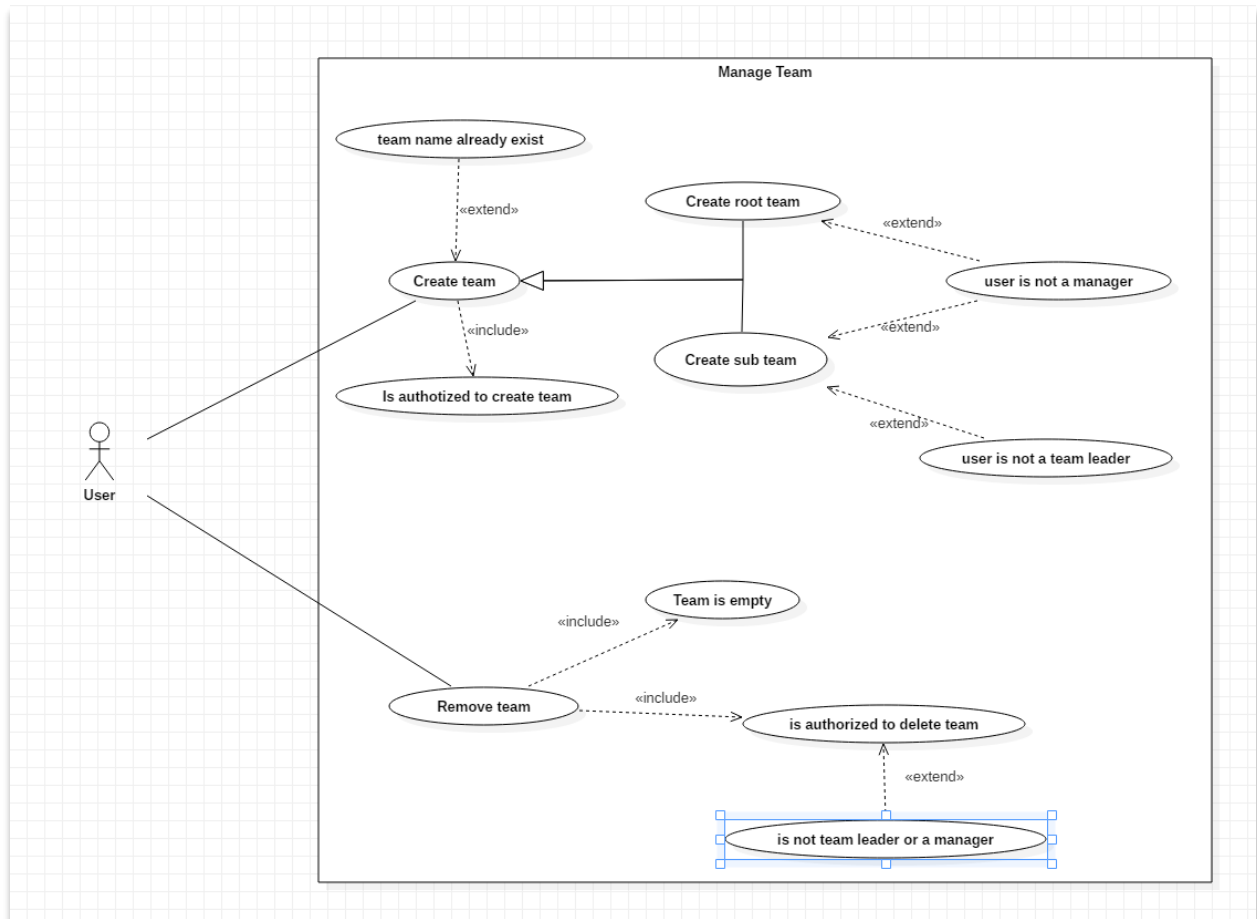


Figure 30create and remove team use case

Manage Employee

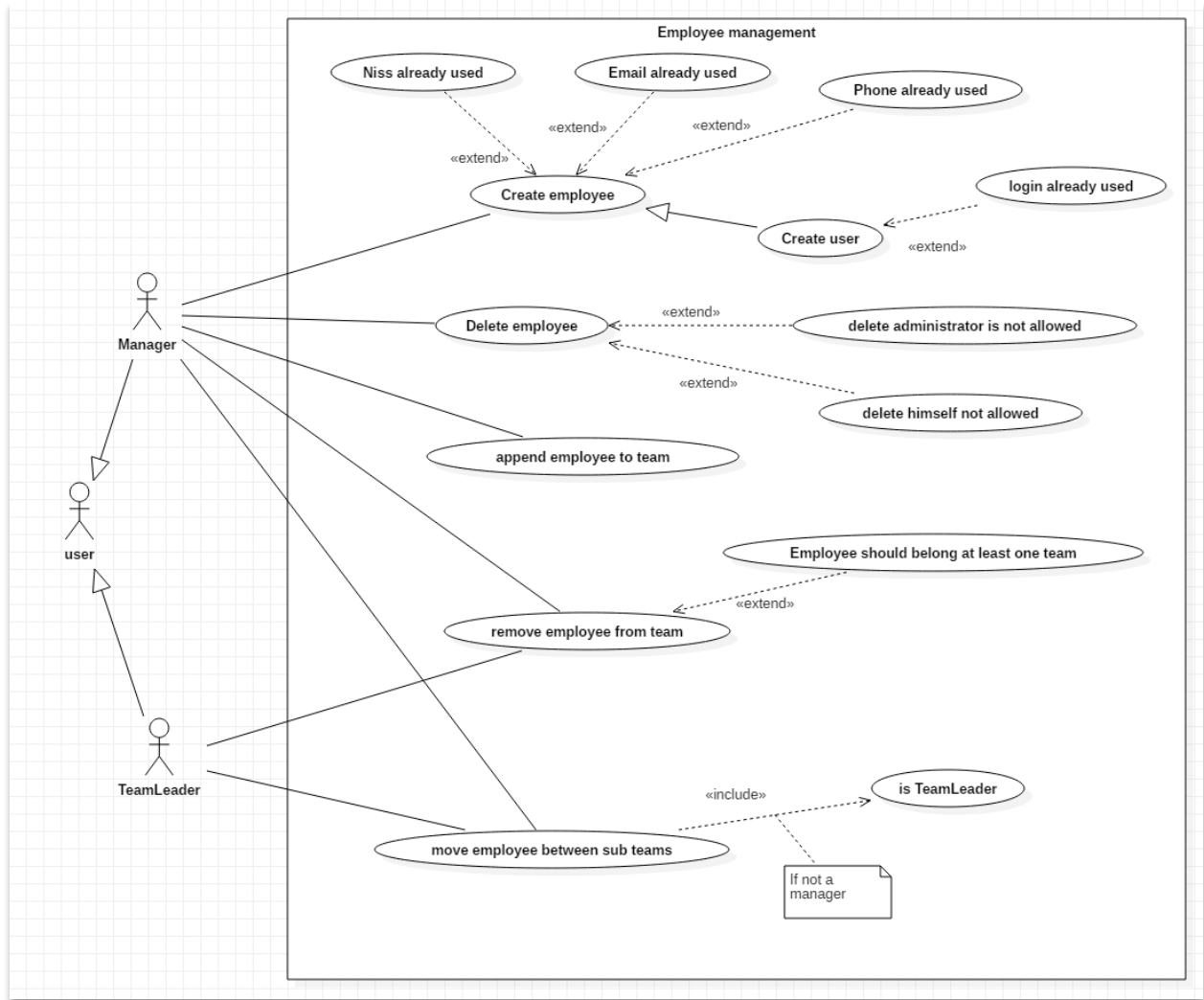


Figure 31 employee management

Class Diagram

Planning diagram

Represent a planning with its utility in the application.

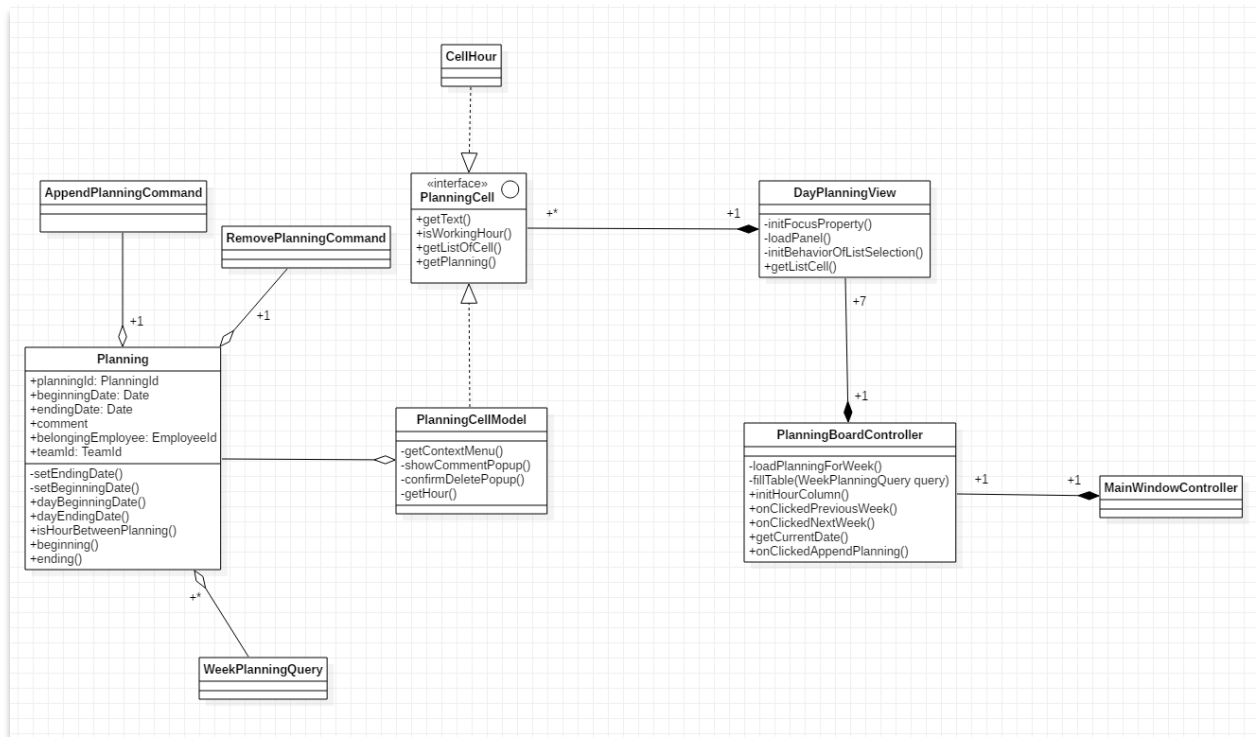


Figure 32planning diagram

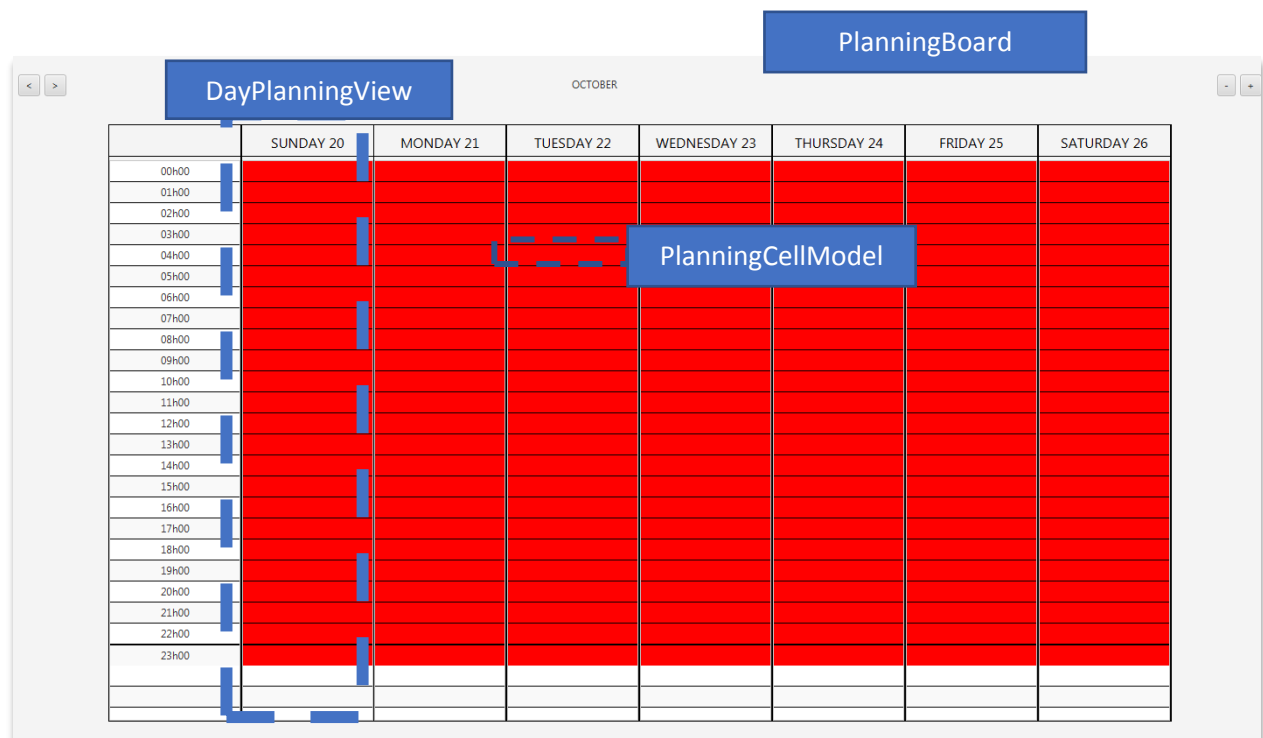


Figure 33view of the planning uml

Search diagram

Represent the generic search features and his implementation for employee

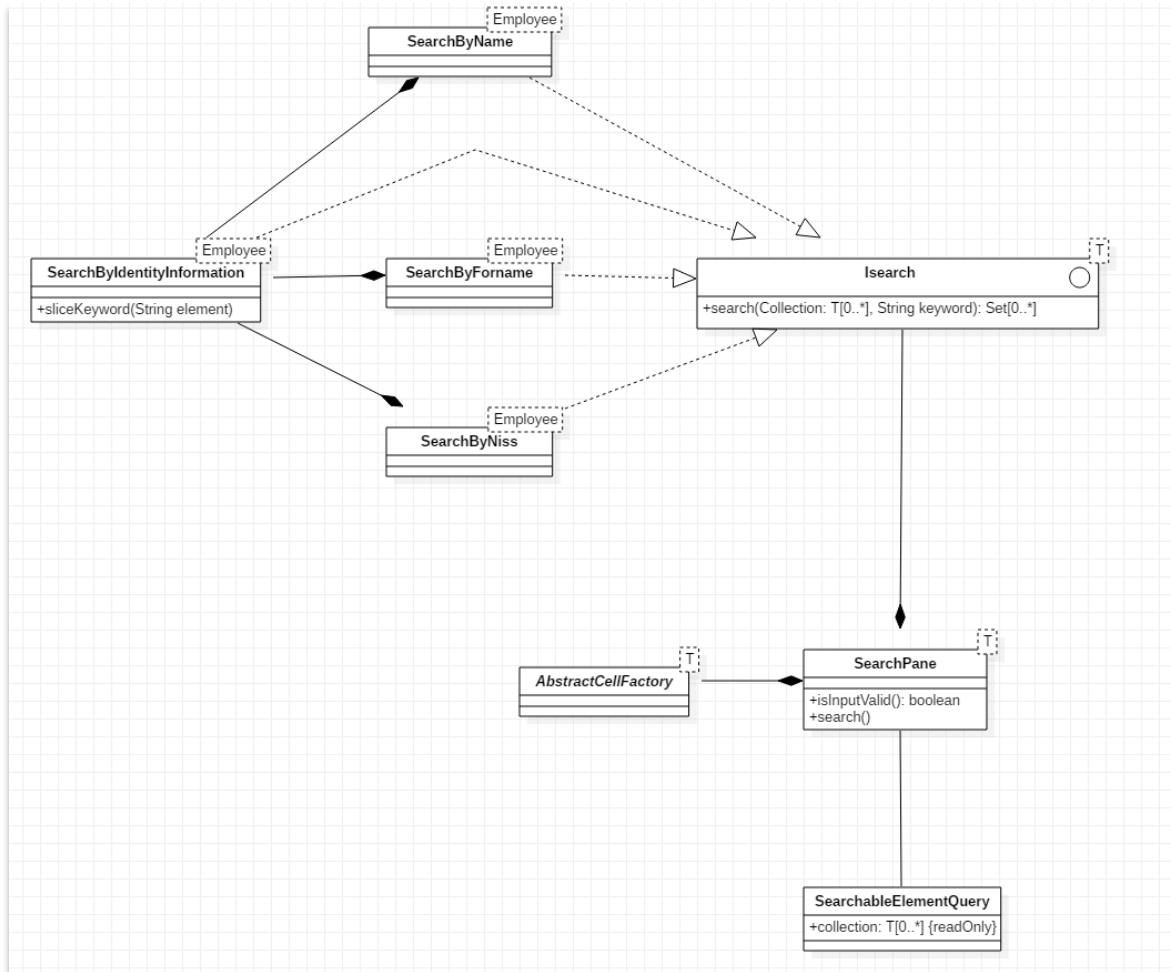


Figure 34search diagram

Thesaurus database

MCD

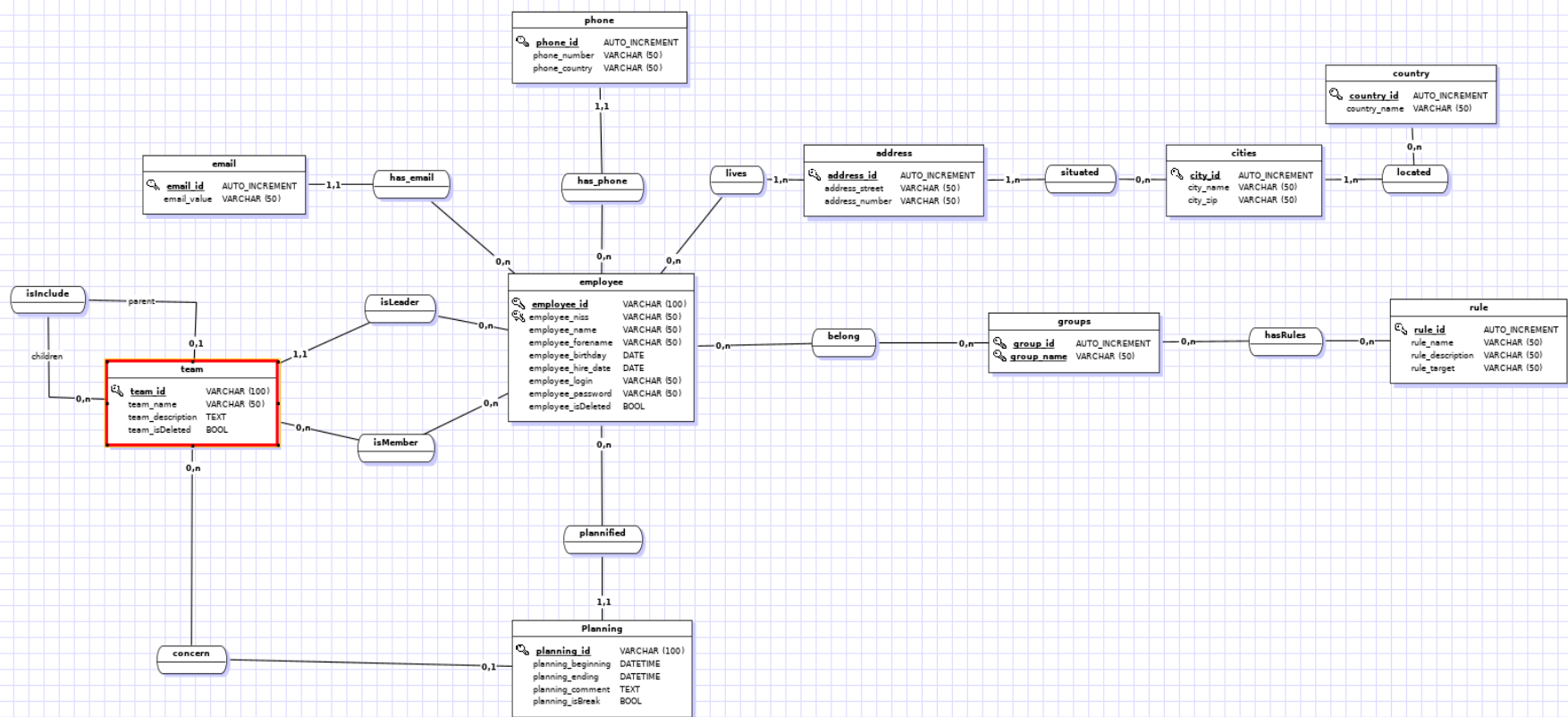


Figure 35mcd

MLD

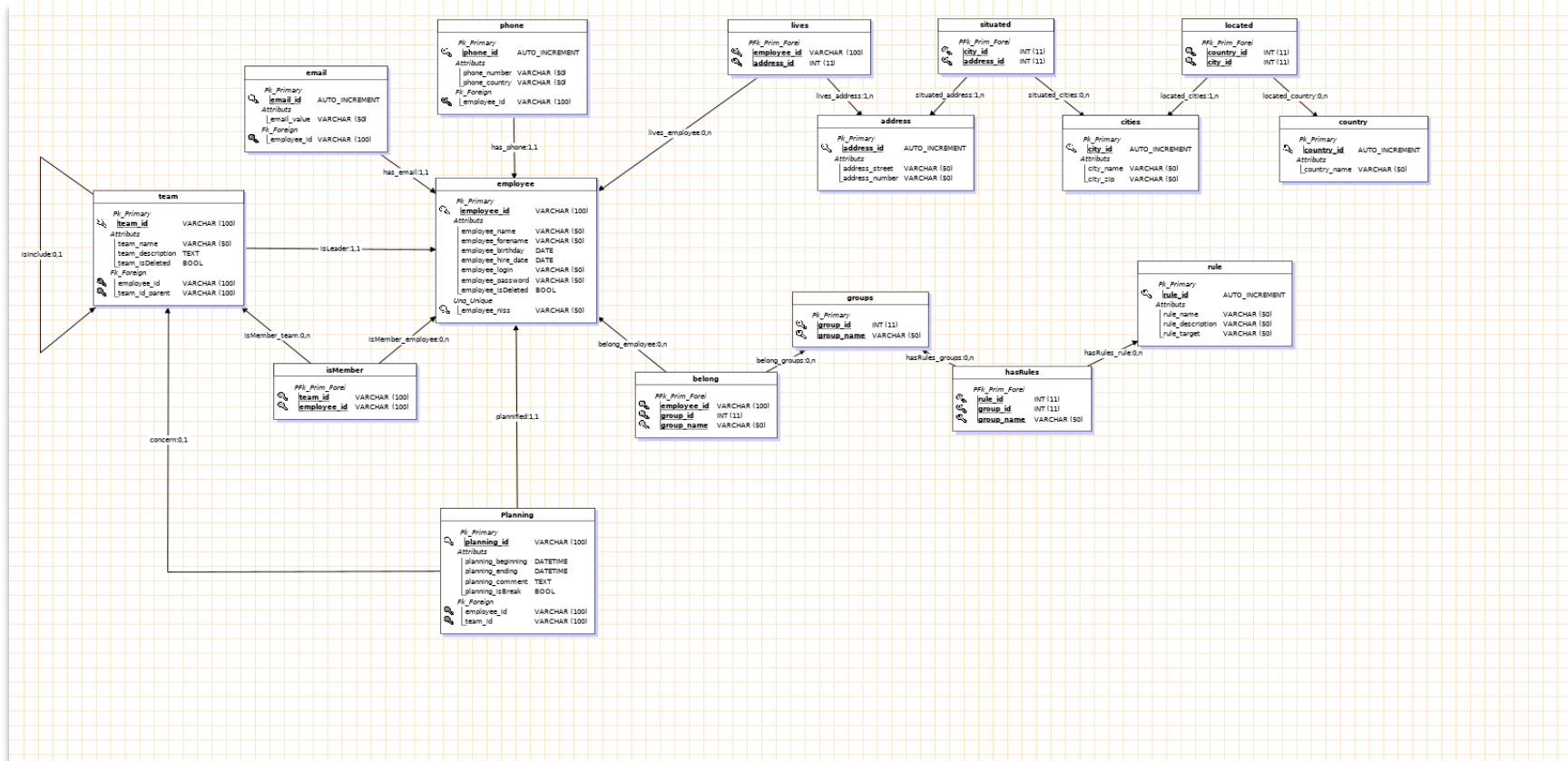


Figure 36mld

Addresses

Country

Represent a country with a name that be unique, a country can have zero or many cities

Country_id	SERIAL auto_incremented	The id generated by the database
Country_name	VARCHAR(50)	The name of the country

Cities

Represent a city with a name and a zip code, the tuple (city,zip) must be unique, a city can belong ONE OR MANY country and has ZERO OR MANY address

city_id	SERIAL auto_incremented	The id generated by database
City_name	VARCHAR(50)	The name of the city
City_zip	VARCHAR(50)	The zip code of the city

Address

Represent the house number and the street where an employee can be live, an address can have ONE OR MANY employee (case where couple works at same work).

Address_id	SERIAL auto_incremented	The id generated by database
Address_street	VARCHAR(50)	The street name
Address_number	VARCHAR(50)	The number of the house

Address example

- 1 | 400 rue du pont | 1300 Wavre | Belgique
- 2 | 72 dreve trevier | 1300 Limal | Belgique

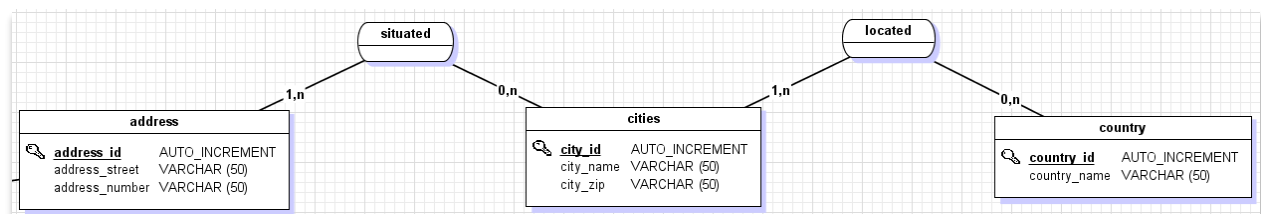


Figure 37mcd address

Contacts

Phone

Represent a phone number with a country code like BE, NL, FR, ... The phone number must be unique

Phone_id	SERIAL auto_incremented	The id generated by database
Phone_number	VARCHAR(50)	The phone number UNIQUE
Phone_country	VARCHAR(50)	The country code

Examples

- 1 | BE | 0478 69 60 60
- 2 | FR | 06 04 12 31 22

Email

Represent an email, the email must be unique

Email_id	SERIAL auto_incremented	The id generated by database
Email_value	VARCHAR(50)	The email UNIQUE

Example

- 1 | anEmail@mail.com

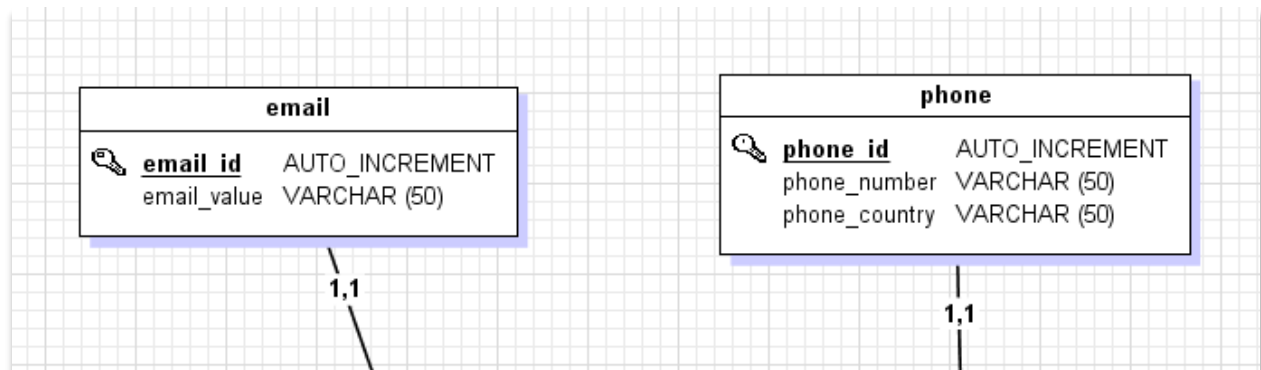


Figure 38mcd contract

Teams

Team

Represent a team where employee can be belonging and where team leader will manage, each team must have ONE AND ONLY ONE Leader, ZERO OR MANY employee, ZERO OR MANY planning and finally a team can have ZERO OR MANY team included (sub team). A team can be deleted only logically.

Team_id	VARCHAR(50)	The id must be managed by the application
Team_name	Varchar(50)	The name of the team (UNIQUE)
Team_description	TEXT	A description of the team
Team_isDeleted	BOOL	Represent if the team is deleted

Example:

- UUID-01 | Technical-team | a team for technic | false
 - isLeader UUID-01 | UUID-EMPLOYEE-1 (team | employee)
 - isMember UUID-01 | UUID-EMPLOYEE-1 (team | employee)
- UUID-02 | Technical-team-division-A | for department A | false
 - IsIncluded UUID-01 | UUID-02 (parent | children)
 - isMember UUID-01 | UUID-EMPLOYEE-2 (team | employee)

The technical team has one sub team with one member (UUID-EMPLOYEE_2), so the employee(2) belongs to technical-team-division-A and technical-team, but the first employee (UUID-EMPLOYEE-1) Not belong to the sub team and he is the team leader

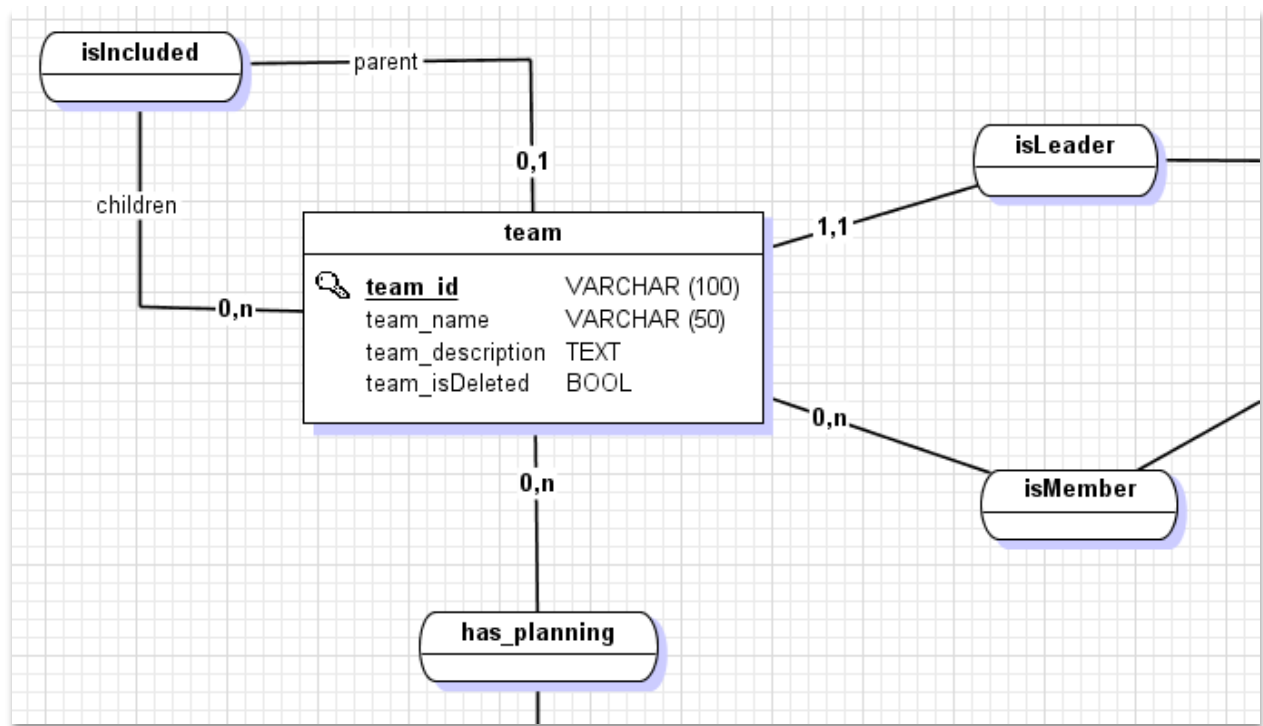


Figure 39mcd team

Planning

planning

Represent planning linked to an employee and a team

Planning_id	VARCHAR(100)	The id must be managed by the application
Planning_beginning	Datetime	Date and time for the beginning
Planning_ending	Datetime	Date and time for ending
Planning_comment	TEXT	A comment for the planning
Planning_isBreak	BOOL	Represent if the planning is a day off

Example:

- UUID-1 | 2019-12-27 08:00:00 | 2019-12-27 19:00:00 | | false |
 - Plannified (UUID-employee-01 | UUID-1)
 - Has_planning (UUID-team-1 | UUID-1)

A planning begin the 27 december 2019 at 8h to 19h for the employee UUID-EMPLOYEE-01 in TEAM UUID-team-01.

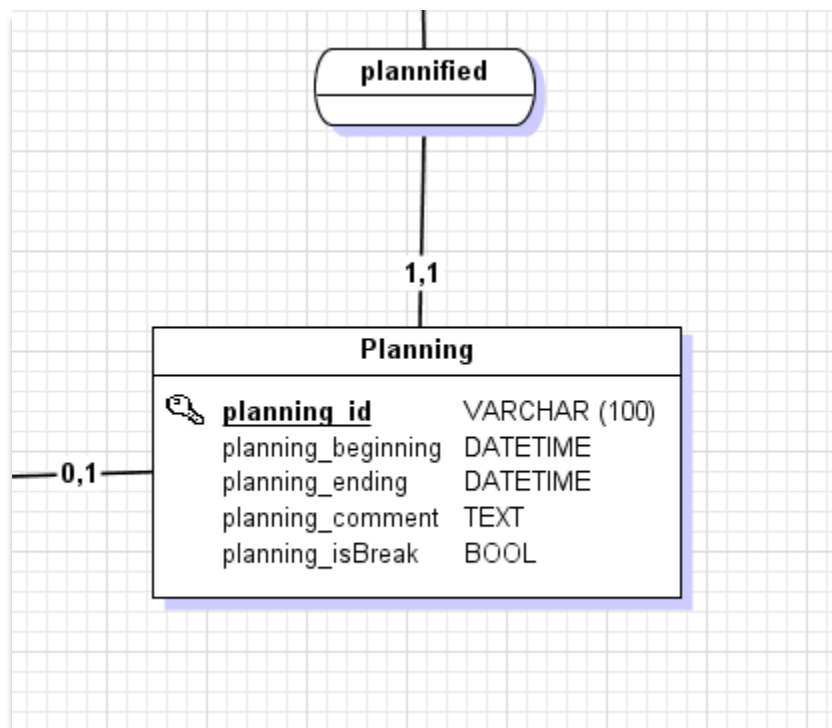


Figure 40mcd planning

Groups authorities

Groups

Represent a group that has ZERO OR MANY rules and has ZERO OR MANY employee. A group represents the authority in the application, so which group employee belongs involve his roles and authorization

Group_id	SERIAL Auto_increment	The id generated by database
Group_name	VARCHAR(50)	The name of the group UNIQUE

Example:

- UUID-GROUP-1 | administrator

Rule

Represent a rule with a name and a target, a rule can belong to ZERO OR MANY groups. The target of a rule can be anything , an employee, a group , a team, A special character '*' for everything

Rule_id	SERIAL auto_incremented	The id generated by database
Rule_name	VARCHAR(50)	The name of the rule
Rule_description	TEXT	Describe the rule
Rule_target	VARCHAR(50)	The target of the rule

Example:

- UUID-RULES-1 | create_employee | rule to create employee | team_manager
 - hasRules UUID-GROUP-1 | UUID-RULES-1

The group administrator can create employee into the team manager.

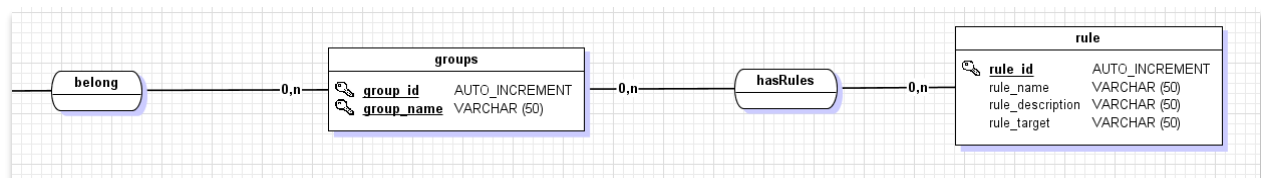


Figure 41MCD group and rule

Employees and user

Employee

Represent an employee who will be an employee, a manager, a team leader... It depends which group he will belong.

Employee_id	VARCHAR(50)	The id must be managed by the application
Employee_niss	VARCHAR(50)	The NISS of the employee
Employee_name	VARCHAR(50)	The name of the employee
Employee_birthday	DATE	The birthday of the employee
Employee_hireDate	DATE	The date when the employee is encoded in application
Employee_login	VARCHAR(50)	The login of the employee nullable
Employee_password	VARCHAR(100)	The password of the employee

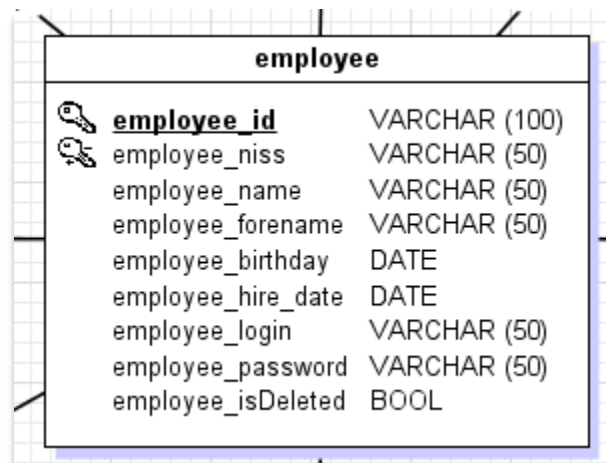


Figure 42mcd employee

User documentation

Big pictures

The screenshot displays the 'Toll Manager' web application. The interface includes a top navigation bar with a search box, a panel selector (Employees, Planning), and a logout button. A left sidebar lists teams and managers. The main content area contains a form for user identity, address, and contact information, along with buttons for team selection and a main panel. Callouts identify specific UI elements: Search Box, Panel selector, Logout button, Sort selector, Buttons control, and Team panel.

Search Box

Panel selector

Logout button

Sort selector

Buttons control

Team panel

Main Panel

Identity:

Name

Forename

Birthday

Niss

Address:

Street: **Number:**

City: **Zip code:**

Country:

Contact:

Email:

Type Team :

picture 1 main panel

Logout

search

Teams

managers

john doe

managers

john doe

root team a

charle bertrand

ellen must

kurts jhonny

meness pierre

root team b

charle bertrand

rouly susan

sub team 1

sub team 2

All in one

-

edit

+

Employees

Planning

Toll Manager

Day selector

OCTOBER

Control Panel

	SUNDAY 13	MONDAY 14	TUESDAY 15	WEDNESDAY 16	THURSDAY 17	FRIDAY 18	SATURDAY 19
00h00							
01h00							
02h00							
03h00							
04h00							
05h00							
06h00							
07h00							
08h00							
09h00							
10h00							
11h00							
12h00							
13h00							
14h00							
15h00							
16h00							
17h00							
18h00							
19h00							
20h00							
21h00							
22h00							
23h00							

picture 2 panel planning

Authorization management

They are 3 kind of user account:

- Team leader
- Manager
- Administrator

Team leader

A team leader has for responsibility his teams (teams attributed by a manager), his role is to manage planning and team management for his employee where he has authority in work.

Roles

- Create, edit or remove sub team
- Arrange employee between his team
- Create, edit or remove planning

Limitation

- You cannot create main team, only sub team.
- You cannot delete a main team, only sub team
- You cannot create, edit or remove employee
- You cannot remove an employee from a main team

Manager

A manager has all rules of a team leader with the possibility to create edit and remove employee and main team.

Roles

- Create, edit or delete employee, team leader and manager

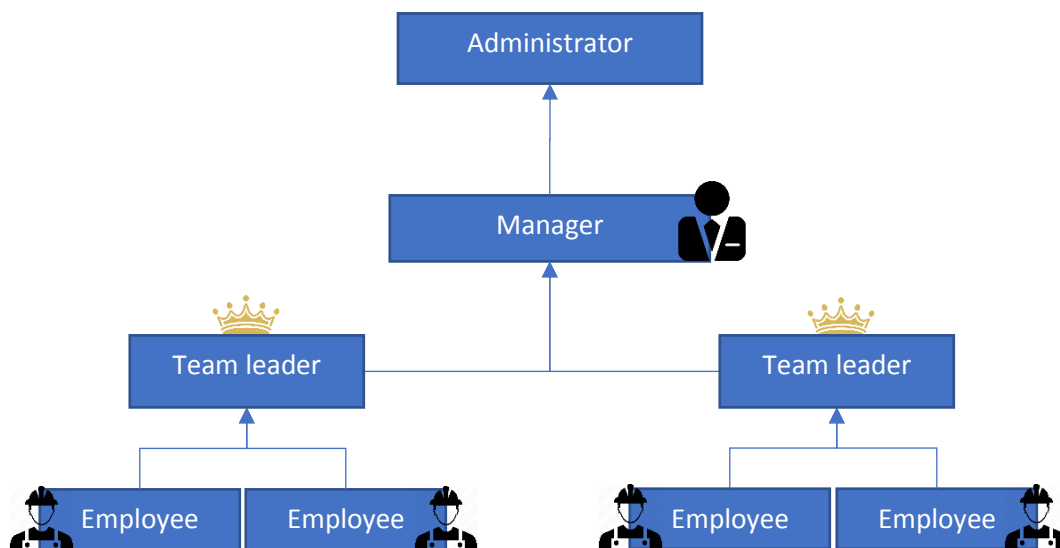
- Create, edit or delete main team
- Assign a leader to a team
- Assign employee to teams

Limitation

- Remove an employee from team if he hasn't team (dangling employee)
- Delete other managers and himself
- Delete or edit administrator

Administrator

An administrator has all rules, he can do everything except one thing delete himself, For the whole application there is only one administrator account.

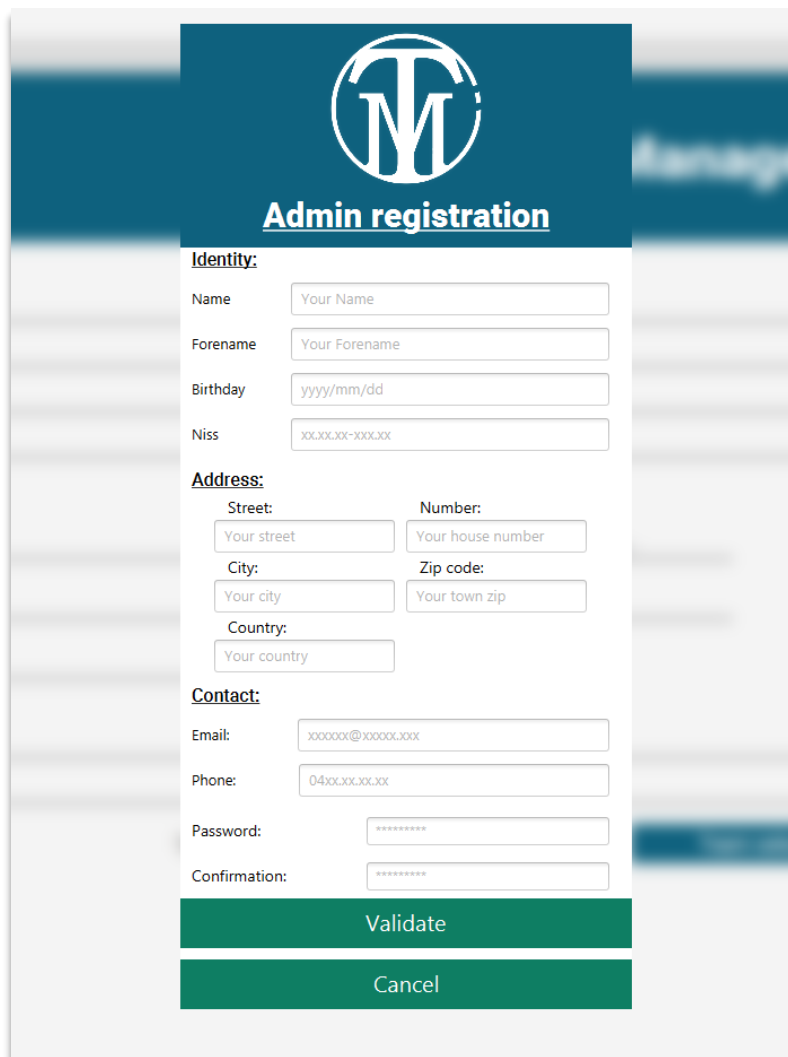


picture 3 organizational chart

Application launch

First launch

When the application is launched for the first time, the administrator form appears, you need to fill it.



Admin registration

Identity:

Name:

Forename:

Birthday:

Niss:

Address:

Street: Number:

City: Zip code:

Country:

Contact:

Email:

Phone:

Password:

Confirmation:

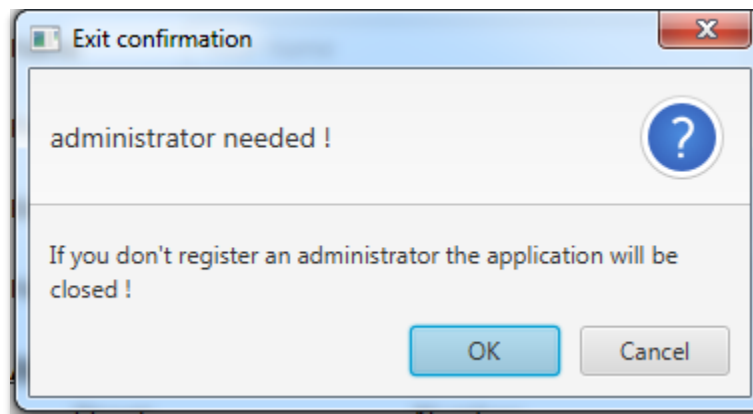
Validate

Cancel

picture 4 the administrator form

When the form is filled correctly the application will be initialized for the first launch and you can sign in with the account *admin* and the password you chosen in the form.

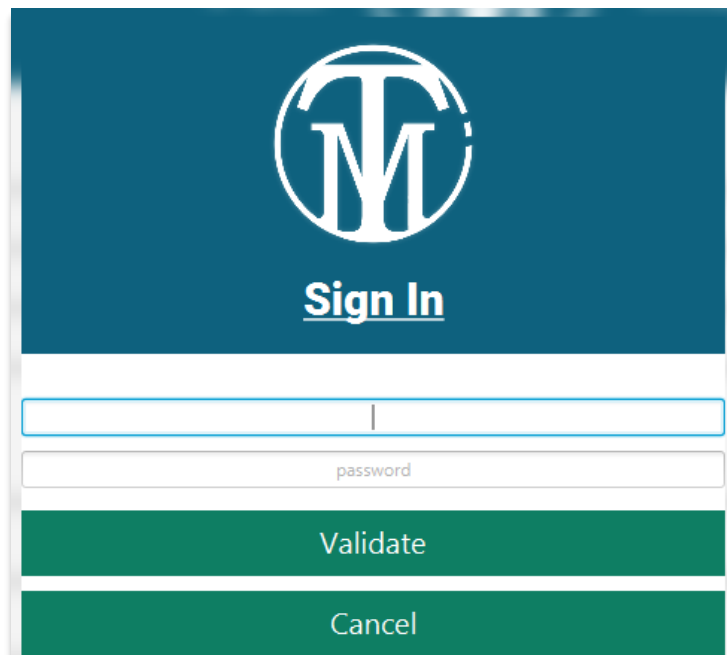
If you don't fill the administrator form then the application cannot be launched, an information popup indicated that.



picture 5 exit administrator form confirmation

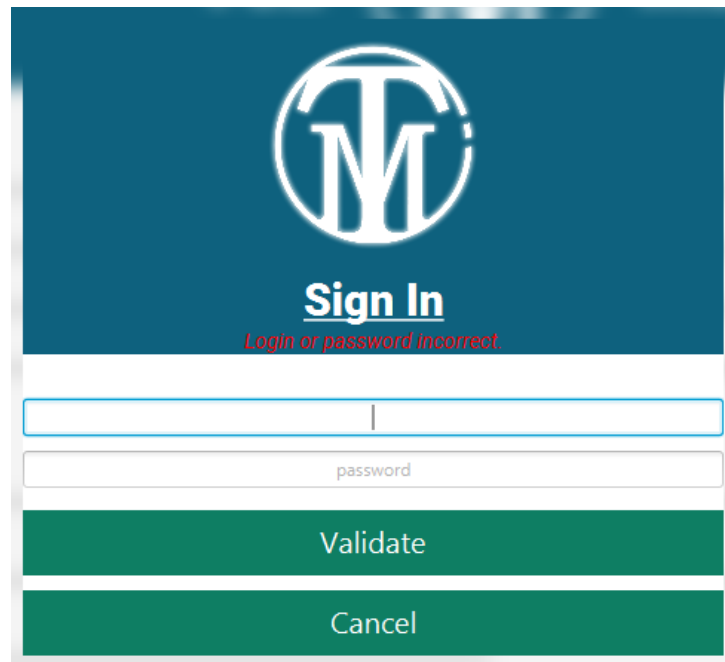
Authentication

When the application is correctly initialized then the sign in form appears.

A sign-in form with a dark teal header. The header contains a white circular logo with the letters "TM" and the text "Sign In" in white. Below the header is a white input field with a vertical line in the center. Underneath is a grey input field with the placeholder text "password". At the bottom, there are two green buttons: "Validate" and "Cancel".

picture 6 sign in form

When the login or the password is incorrect then a red message will be displayed just below the title 'sign in' and fields are cleaned.



The image shows a 'Sign In' form with a dark blue header. At the top center is a white circular logo containing the letters 'TM'. Below the logo, the text 'Sign In' is displayed in white, bold font. Underneath 'Sign In', a red error message 'Login or password incorrect' is visible. The form has two input fields: the first is empty and has a blue border, and the second contains the placeholder text 'password' and has a grey border. Below the input fields are two green buttons: 'Validate' and 'Cancel'.

picture 7sign in form with error

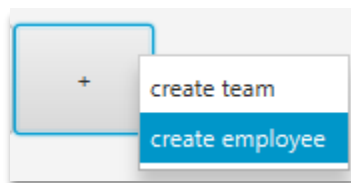
Employee management

When you are logged like a manager or administrator you can have an access to the control button panel situated at left bottom on the frame.



picture 8 control button panel

Create an employee



picture 9 create employee

When you click on “+” button, a little menu will be showed, then click on ‘*create employee*’ to unlock the creation employee form.

A form for creating an employee. It has three main sections: 'Identity' with fields for Name, Forename, Birthday, and Niss; 'Address' with fields for Street, City, Country, Number, and Zip code; and 'Contact' with fields for Email and Phone. At the bottom, there are dropdown menus for 'Type' (set to 'employee') and 'Team' (set to 'root team b'). Below these are two buttons: 'Validate' (green) and 'Cancel' (red).

picture 10 employee form unlocked

When a field is correctly filled then it becomes green, else it becomes red.

Two input fields. The first field contains the text 'doe' and has a green border, indicating it is correctly filled. The second field contains the text '9999/99/99' and has a red border, indicating it is not correctly filled.

Some Fields that need to be unique are directly connected with the database:

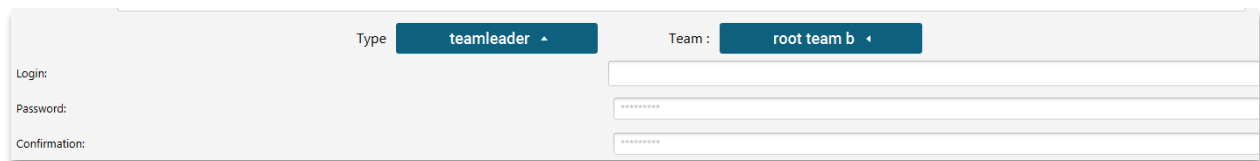
- Niss
- Email
- Phone
- Login

If the user types data already existing in the database then the field becomes red.

If one field are red the whole form becomes invalidated and you cannot register the employee.

Create a team leader or a manager

When you want to append a team leader or a manager you need to select the desired type of employee in the selected menu 'type' bottom the form, then the user form appears.



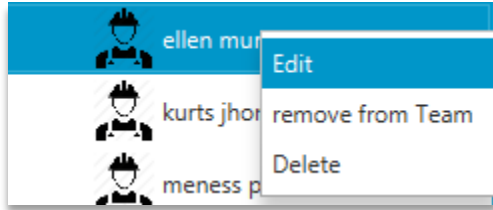
The screenshot shows a user registration form. At the top, there are two dropdown menus: 'Type' with 'teamleader' selected and 'Team' with 'root team b' selected. Below these are three input fields: 'Login:', 'Password:', and 'Confirmation:'. The 'Password:' and 'Confirmation:' fields have masked text (dots) visible.

picture 11the user form

Assign team leader to a team

When you assign a team leader to a team, a popup ask you if you want to replace the current leader , if you valid the chosen then the old team leader stay on team but like a employee.

Edit an employee



To edit an employee, you need to do a right click on his name on the left panel and select 'edit'.

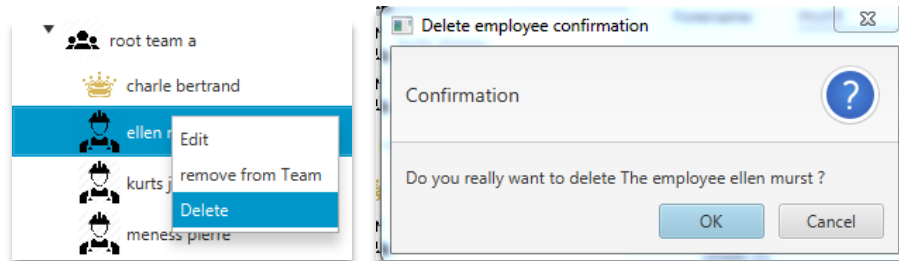
When you have clicked then the form will be filled with the employee information and becomes editable.

A screenshot of an employee edit form. The form is divided into several sections: 'Identity:', 'Address:', and 'Contact:'. The 'Identity:' section has four input fields: 'Name' (filled with 'ellen'), 'Forename' (filled with 'murst'), 'Birthday' (filled with '1999/12/12'), and 'Niss' (filled with '11.11.11-333.11'). The 'Address:' section has six input fields: 'Street:' (filled with 'street 01'), 'Number:' (filled with '12'), 'City:' (filled with 'liege'), 'Zip code:' (filled with '4000'), 'Country:' (filled with 'belgium'). The 'Contact:' section has two input fields: 'Email:' (filled with 'fakeemail@gmail.com') and 'Phone:' (filled with '0478.62.60.60'). Below the input fields, there are two dropdown menus: 'Type' (set to 'employee') and 'Team' (set to 'root team b'). At the bottom of the form, there are two buttons: 'Validate' (green) and 'Cancel' (red).

picture 12form editable with employee information

Delete an employee

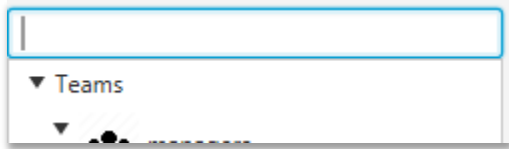
To delete an employee, you need to do a right click on his name on the left panel and select “delete”, the employee will be deleted from the application but not from the database



There are 2 cases where the employee cannot be deleted:

1. The user tries to delete himself
2. The employee to del is the administrator

Search an employee



picture 13searchbox

If you want to search an employee among teams where you have authorities (teams for manager and administrator) you can write the name or forename of the employee searched in the searchbox

situated on the top of the left panel.

The search box needs minimal 2 character to beginning the searching, when employees were founded then they appear below the field.

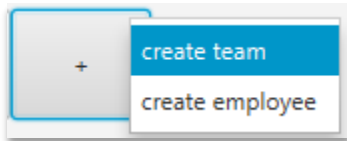


picture 14search result

Team management

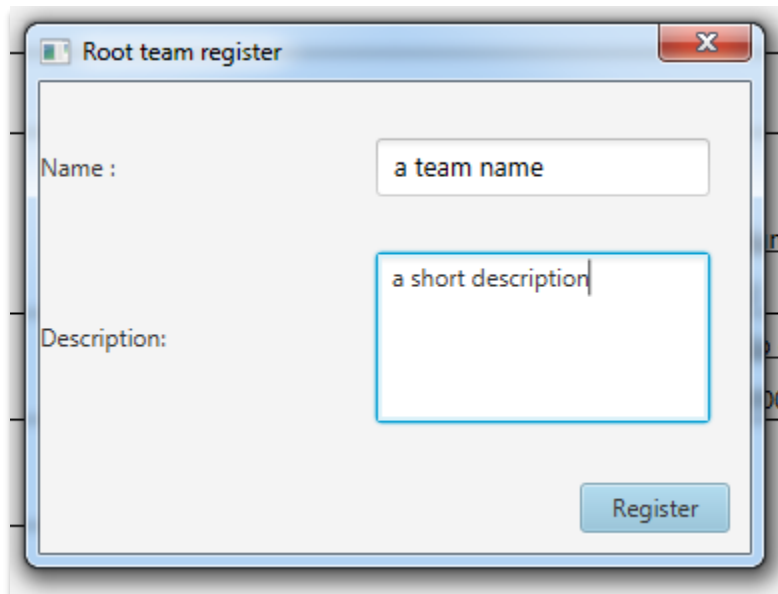
When you are logged like a manager or administrator you can have an access to the control button panel situated at left bottom on the frame.

Create a root team



picture 15 create team

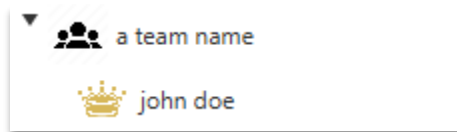
When you click on “+” button, a little menu will be showed, then click on ‘*create team*’ to show the creation team popup.

A screenshot of a 'Root team register' popup window. The window has a title bar with a close button. Inside, there are two input fields: 'Name :' with the placeholder text 'a team name' and 'Description:' with the placeholder text 'a short description'. A 'Register' button is located at the bottom right.

picture 16 team creation popup

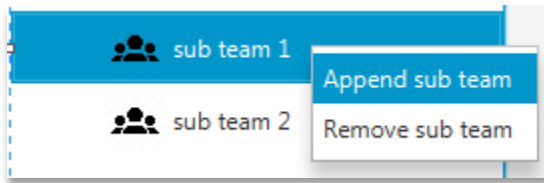
A team must have at least 3 characters length and the name must be unique.

When the team has been created then it appears on the left panel and the creator of the team (you, connected user) are the team leader.



picture 17 team created you're the leader

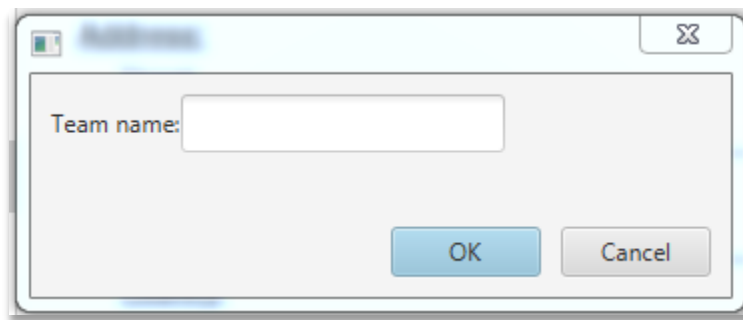
Create a sub team



picture 18 append sub team menu

To create a sub team to a main team, you need to do a right click on the team or on the sub team and select 'append sub team'

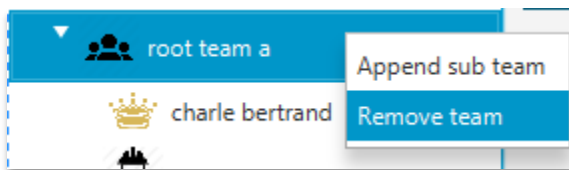
When you have clicked on the '*append sub team*' then a popup to select the name of the sub team appears



picture 19 sub team popup

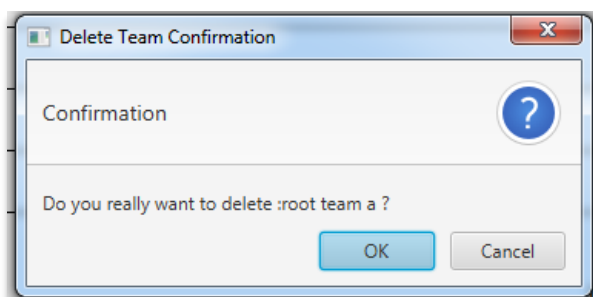
when you click on 'ok' the sub team will be appended to the parent team.

Remove a main team

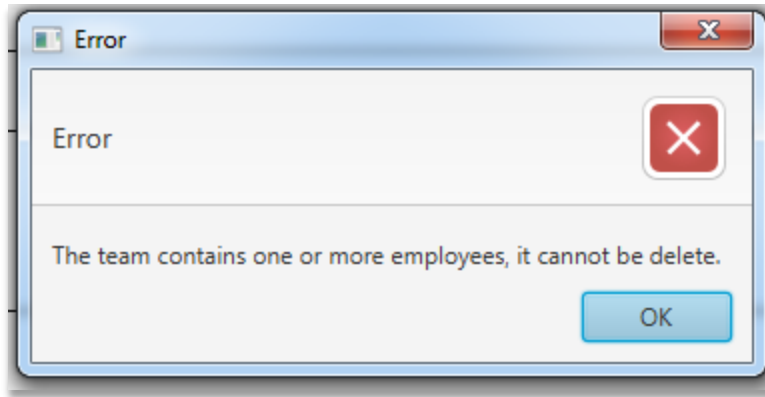


picture 20 remove team

To remove a main team, you need to do a right click on the team and select 'Remove team'.

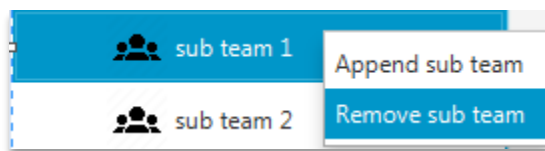


Then a confirmation popup appears, if the team contains any other member than the team leader then the team cannot be deleted, you must place employee to other team.



picture 21error delete team

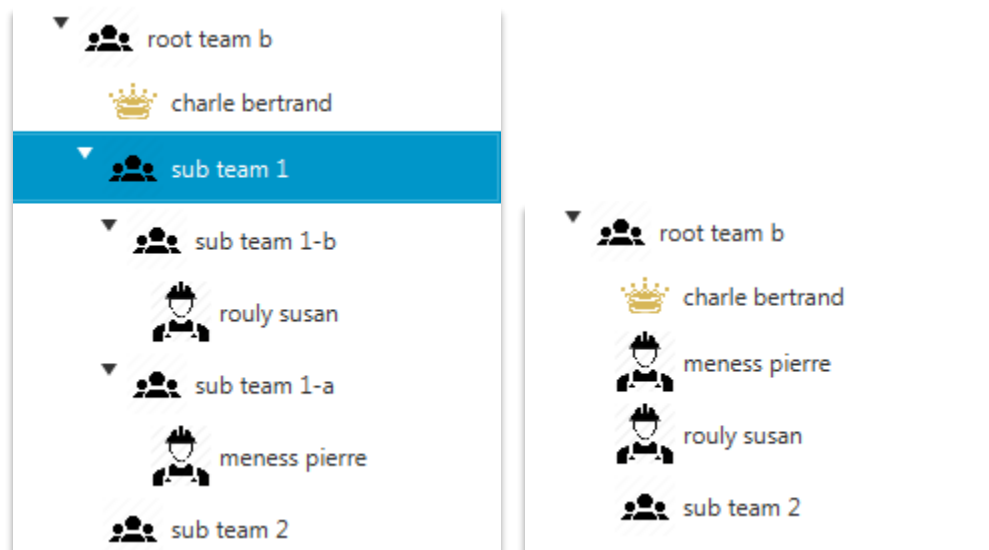
Remove a sub team



picture 22remove sub team

To remove a main team, you need to do a right click on the team and select 'Remove sub team'.

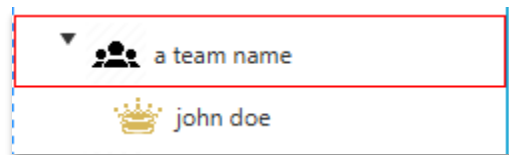
Like the main team, a popup appears to confirm, but if the sub team has employee it moves every employee to the parent team



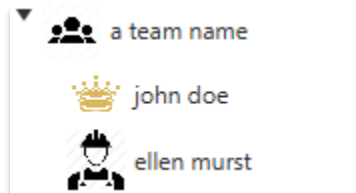
picture 23 Left before delete sub team, right after

Append employee to team

If you want to append an employee to a team, you need to click on the employee stay clicked and drag him to the team, a red rectangle confirms the employee can be append to the team.



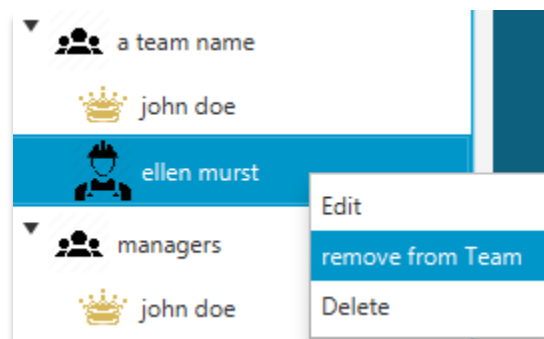
picture 24employee can be dropped here



picture 25the employee has been added the to team

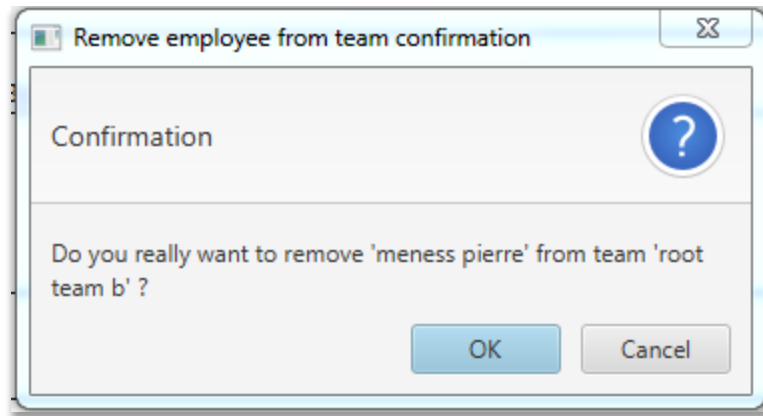
Remove employee to team

If you want to remove an employee to a team, you need to right click on the employee and to select '*remove from team*'



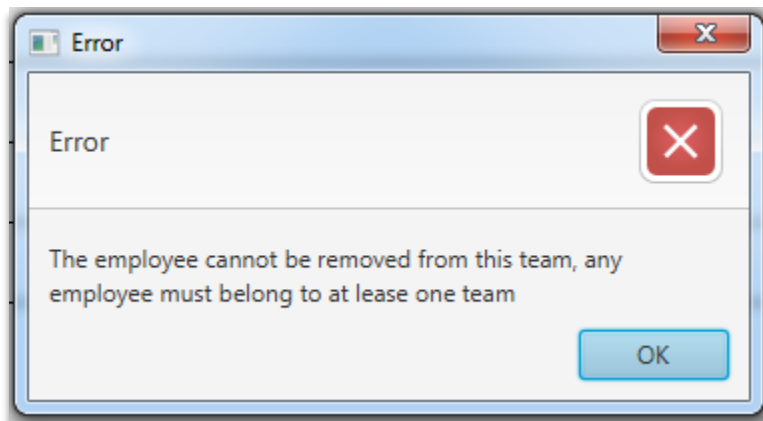
picture 26remove from team

A confirmation popup will be displayed to confirm your action



picture 27Confirm remove employee from team

if the employee not belong to at least one team, he cannot be removed, that means any employee must belong to one team.



picture 28error remove employee from team


Planning management

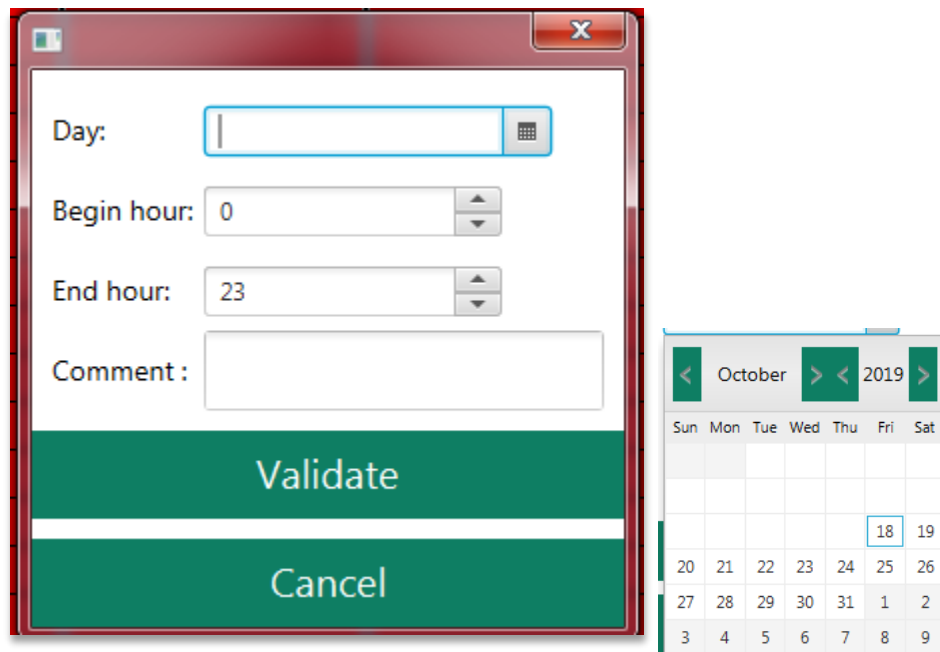
When you have selected an employee the planning panel selector situated on the becomes available.



picture 29 left not available, right available

Append a day planning

 You can click on the '+' button to append a planning for the employee selected, then a popup with the planning form creator appears.



picture 30 left Popup planning creator form, right day selector

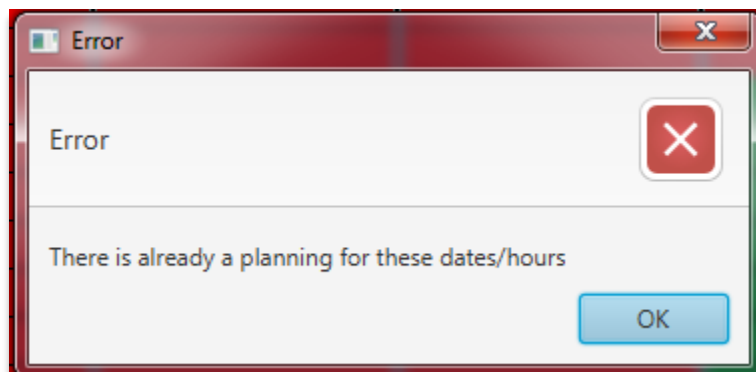
If you click on the calendar icon on the right of field *Day*, then the day selector appears and you can select a day superior or equals to today. Indeed, it is impossible to create, edit or remove a planning when the ending date is outdated.

When the form is filled correctly you'll see your planning on the calendar.

	SUNDAY 13	MONDAY 14	TUESDAY 15	WEDNESDAY 16	THURSDAY 17	FRIDAY 18	SATURDAY 19
00h00							
01h00							
02h00							
03h00							
04h00							
05h00							
06h00							
07h00							
08h00							
09h00							
10h00							
11h00							
12h00							
13h00							
14h00							
15h00							
16h00							
17h00							
18h00							
19h00							
20h00							
21h00							
22h00							
23h00							

picture 31planning added

If you try to append a planning where a planning already present a popup information inform you that's not possible, cause there it's possible to have only one planning between two hours.



picture 32error to insert already present planning

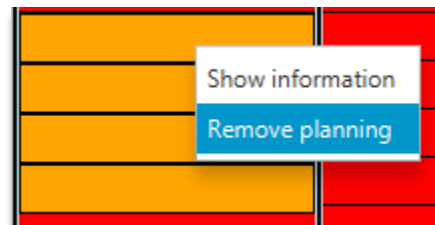
But you can have many planning on one day.

	SUNDAY 13	MONDAY 14	TUESDAY 15	WEDNESDAY 16	THURSDAY 17	FRIDAY 18	SATURDAY 19
00h00							
01h00							
02h00							
03h00							
04h00							
05h00							
06h00							
07h00							
08h00							
09h00							
10h00							
11h00							
12h00							
13h00							
14h00							
15h00							
16h00							
17h00							
18h00							
19h00							
20h00							
21h00							
22h00							
23h00							

picture 33many planning for one day

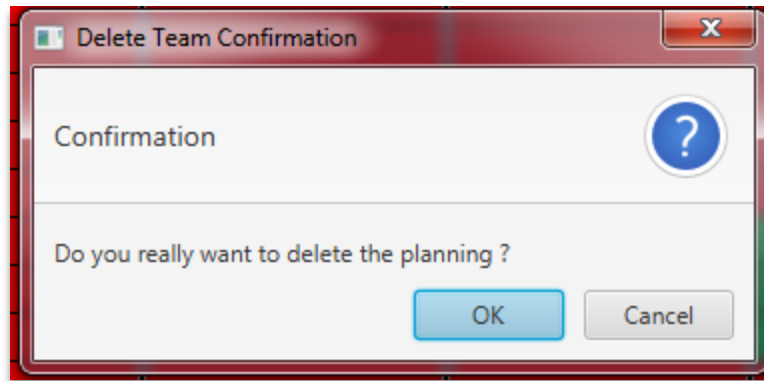
Remove a planning

If you want to remove a planning, first of all the planning must be not outdated, if it's the case then you can do a right click on the planning, it becomes orange that mean the planning is selected, and click on 'remove planning'.



picture 34remove planning menu

Then a confirmation popup will be display to confirm the suppression.



picture 35confirm planning suppression

Navigate on week



You can navigate on weeks with the button '<' and '>' situated on the top left, you'll see day column changed.

	SUNDAY 20	MONDAY 21	TUESDAY 22	WEDNESDAY 23	THURSDAY 24	FRIDAY 25	SATURDAY 26
00h00							

picture 36Days column

Show information



If you do a right click on the planning and select the 'Show information' menu a popup with complete information about the planning selected appears.

