```java
package de.hawai.bicycle_tracking.server.astcore.customermanagement;

import static org.assertj.core.api.Assertions.assertThat;
import static org.junit.Assert.fail;

import java.util.Date;

import javax.annotation.Resource;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.TestExecutionListeners;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.support.AnnotationConfigContextLoader;
import org.springframework.test.context.support.DependencyInjectionTestExecutionListener;
import org.springframework.test.context.support.DirtiesContextTestExecutionListener;
import org.springframework.test.context.transaction.TransactionConfiguration;
import org.springframework.test.context.transaction.TransactionalTestExecutionListener;
import org.springframework.transaction.annotation.Transactional;

import de.hawai.bicycle_tracking.server.AppConfig;
import de.hawai.bicycle_tracking.server.security.HawaiAuthority;
import de.hawai.bicycle_tracking.server.utility.value.Address;
import de.hawai.bicycle_tracking.server.utility.value.EMail;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(loader = AnnotationConfigContextLoader.class, classes = AppConfig.class)
@Transactional(noRollbackFor = Exception.class)
@TestExecutionListeners(listeners = {DependencyInjectionTestExecutionListener.class,
        DirtiesContextTestExecutionListener.class, TransactionalTestExecutionListener.class})
@TransactionConfiguration(transactionManager = "transactionManager", defaultRollback = true)
public class UserIT {

    private static final String NAME = "Name";
    private static final String FIRST_NAME = "FirstName";
    private static final EMail E_MAIL_ADDRESS = new EMail("foo@bar.com");
    private static final Address ADDRESS = new Address("Foostreet", "1a", "Barheim", "DC", "1337", "Germany");
    private static final Date BIRTHDATE = new Date(0);
    private static final String PASSWORD = "TestingPassword";

    @Resource(name = "IUserDao")
    private IUserDao userDao;

    private User user;

    @Before
    public void setup() {
        user = new User(NAME, FIRST_NAME, E_MAIL_ADDRESS, ADDRESS, BIRTHDATE, PASSWORD, HawaiAuthority.USER);
        userDao.save(user);
    }

    @Test
    public void getOneByID_UserExists_UserCanBeFoundByID() throws Exception {
        User userFromDB = userDao.getOne(user.getId());
        assertThat(user).isEqualTo(userFromDB);
    }

    @Test
    public void getOneByID_UserExists_UserAttributesAreEqual() throws Exception {
        User userFromDB = userDao.getOne(user.getId());
        assertThat(user).isEqualTo(userFromDB);
        assertThat(user.getName()).isEqualTo(userFromDB.getName());
        assertThat(user.getFirstName()).isEqualTo(userFromDB.getFirstName());
        assertThat(user.getMailAddress()).isEqualTo(userFromDB.getMailAddress());
        assertThat(user.getAddress()).isEqualTo(userFromDB.getAddress());
        assertThat(user.getBirthdate()).isEqualTo(userFromDB.getBirthdate());
        assertThat(NAME).isEqualTo(userFromDB.getName());
        assertThat(FIRST_NAME).isEqualTo(userFromDB.getFirstName());
```

```
72          assertThat(E_MAIL_ADDRESS).isEqualTo(userFromDB.getMailAddress());
73          assertThat(ADDRESS).isEqualTo(userFromDB.getAddress());
74          assertThat(BIRTHDATE).isEqualTo(userFromDB.getBirthdate());
75          assertThat(PASSWORD).isEqualTo(userFromDB.getPassword());
76      }
77
78      @Test
79      public void getByMail_UserDoesntExit_UserCantBeFound() throws Exception {
80          assertThat(userDao.getByMailAddress(new EMail("foobar@bar.gov")).isPresent()).
    isFalse();
81      }
82
83      @Test
84      public void save_UserWithSameMailExists_UserCantBeSaved() throws Exception {
85          try {
86              userDao.saveAndFlush(new User("other name",
87                      "other first name", E_MAIL_ADDRESS,
88                      new Address("A", "B", "C", "D", "E", "F"),
89                      new Date(42),
90                      "Other Password", HawaiAuthority.USER));
91              fail("DataIntegrityViolationException expected because test tries to save a
    user with an already existent email address.");
92          } catch (DataIntegrityViolationException e) {
93              // do nothing
94          }
95      }
96
97      @Test
98      public void save_UserWithoutBirthday_UserCantBeSaved() throws Exception {
99              userDao.save(new User("other name",
100                     "other first name", new EMail("somerandom@mail.com"),
101                     new Address("A", "B", "C", "D", "E", "F"),
102                     null,
103                     "Other Password", HawaiAuthority.USER));
104     }
105 }
106
```