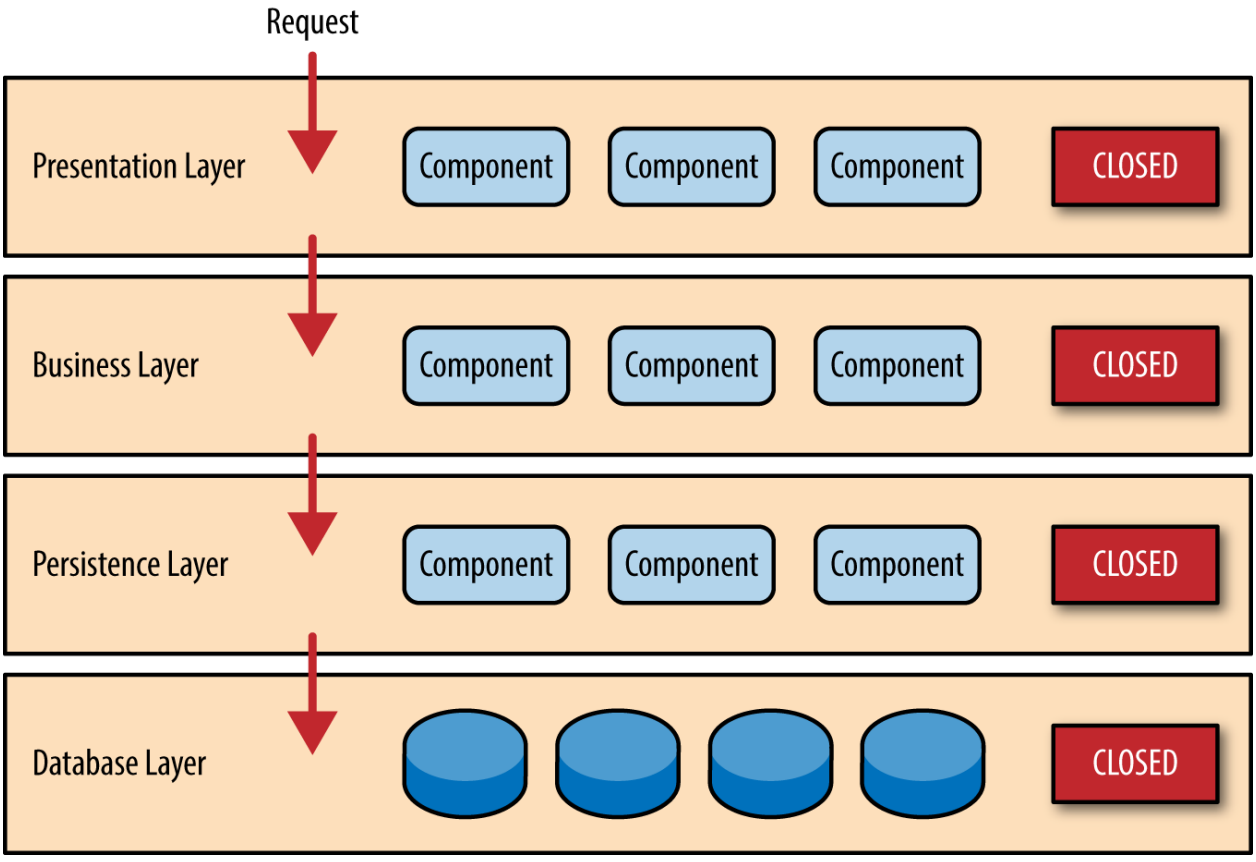
Assignment 1

Simina Radu Alexandru

1. **Layered architecture**

For this assignment I have chosen to work under a layered architecture pattern. Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, my presentation layer is responsible for handling all user interface, whereas a business layer would be responsible for executing specific business rules associated with the request. One of the powerful features of the layered architecture pattern is the *separation of concerns* among components. Components within a specific layer deal only with logic that pertains to that layer. For example, components in the presentation layer deal only with presentation logic, whereas components residing in the business layer deal only with business logic. This type of component classification makes it easy to build effective roles and responsibility models into your architecture, and also makes it easy to develop, test, govern, and maintain applications using this architecture pattern due to well-defined component interfaces and limited component scope.



In the case of my project, the presentation level consists in the **resources.templates** package. Here I have created all the GUI classes using HTML5/CSS/JavaScript and Thymeleaf. The only attributes of these are to offer a clean path of communication between the user and the software. There are multiple templates, all of them self-explanatory, with relevant labels, and pop-up messages.

The Business layer is represented in my project as the **controller**. In here, one can find all the actions performed when a certain input is received from the presentation layer (GUI). Also, inside this package, there have been made validations for the input data.

The persistence layer is situated in the **service** package, where each entity has it`s specific operations. In this layer, I have established the connection with the database, created and executed queries, using JpaRepository

The other packages involved in the project are: **entity** which will contain the model of the objects used during the development of the application. In the **utilities** package, we have multiple classes, one of which is responsible for composing and sending emails.

Here we have the **Authentication** class that will retrieve the username of the user authenticated on the server. Then we have 2 event listeners for successful and failed login attempts, that are linked with the security part. The **CheckDateInRange** class has 2 methods, 1 for validating that the dates are in a correct order, and the other which checks if a third date is situated between the first two. The **ComputeAge** class, has a method which takes as an argument the date of birth of a user and returns ana int symbolizing his age. The **Exchange** stores the exchange rates of the different currencies available. The magic is in the **SecurityConfiguration** class which configures Spring Security. Here we have 2 implementations of the configure method inherited from **WebSecurityConfigurerAdapter.** The first one tells Spring Security where to look for data when a user tries to login, whether it is via email or via username. The second implementations has the following roles:

- To authorize requests only for the **antMatched** URLs
- To add our custom login page as the default login page for Spring Security
- To redirect the user in both cases: of success and failure
- To redirect any other attempt to the login page.

Also, there is a Bean containing the password encryption mechanism used.

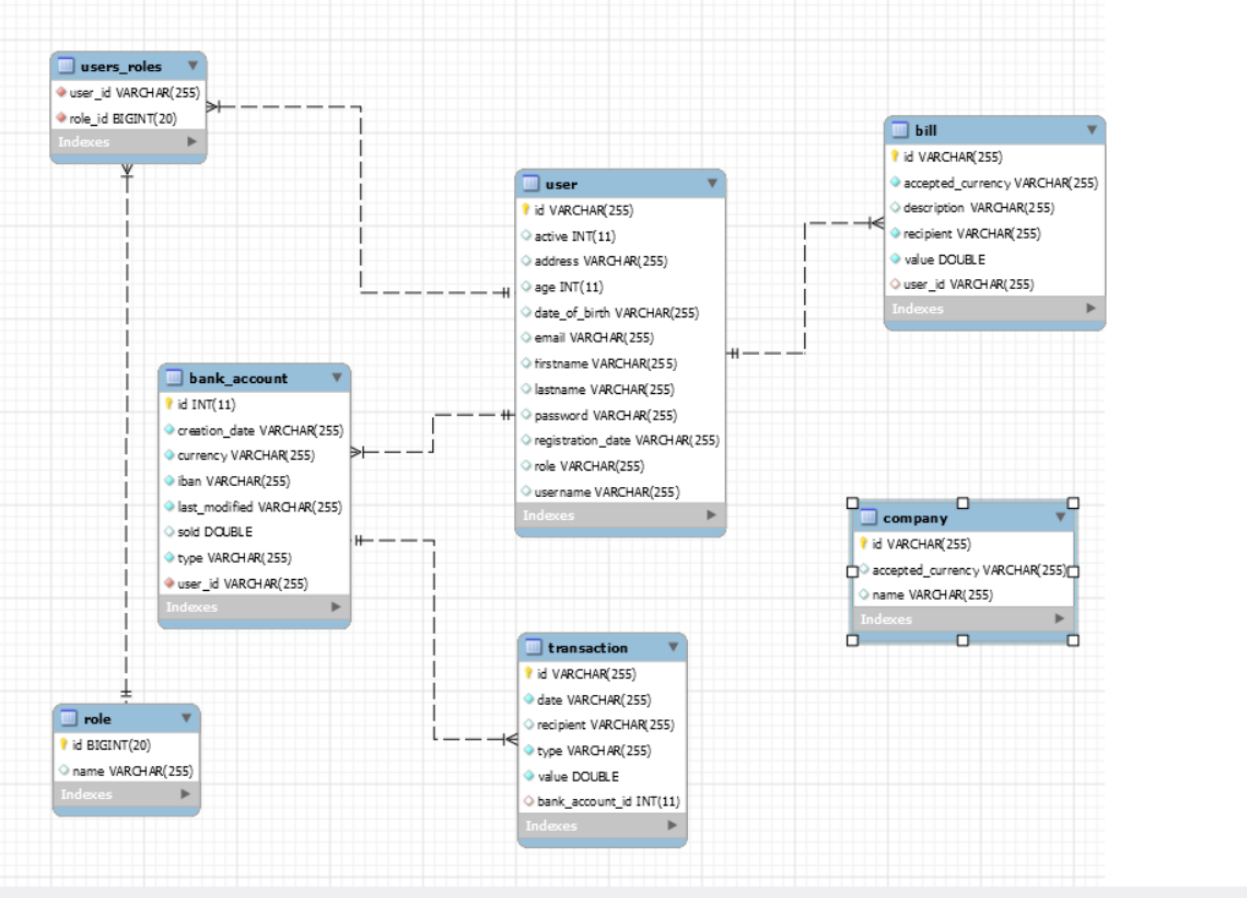
Another package that can be found in the implementation is the DTO package which

contains the form package as well. This 2 packages are used in order to send and receive data between the view and the controller. Using this method we can ensure the fact that sensible data are not sent to our model, thus cannot be stolen – hacked.

2. The database

The database layer. Our database schema, was created locally, manually, using the MySQL workbench application. After that, all the modifications that are brought to the database are made through the Java code, including creating table, inserting values into the table, deleting, updating and so on. However, there is one exception. We had, indeed, to create a manual query and insert it directly into the database via MySQL workbench, and the reason for that was to add an administrator. Since we do not want to offer any user the option to activate his account as an admin, we chose not to offer at all the possibility of adding new administrators through the graphical interface.

Our database model looks like this:



3. Specific MySQL queries:

Some of these queries are hand written, and others will be copy-pasted from the console, given as output the JPA framework.

a) Creating the tables:

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema a2_sd
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema a2_sd
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `a2_sd` DEFAULT CHARACTER SET utf8mb4 ;
USE `a2_sd` ;

-- -----------------------------------------------------
```

```
-- Table `a2_sd`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`user` (
  `id` VARCHAR(255) NOT NULL,
  `active` INT(11) NULL DEFAULT NULL,
  `address` VARCHAR(255) NULL DEFAULT NULL,
  `age` INT(11) NULL DEFAULT NULL,
  `date_of_birth` VARCHAR(255) NULL DEFAULT NULL,
  `email` VARCHAR(255) NULL DEFAULT NULL,
  `firstname` VARCHAR(255) NULL DEFAULT NULL,
  `lastname` VARCHAR(255) NULL DEFAULT NULL,
  `password` VARCHAR(255) NULL DEFAULT NULL,
  `registration_date` VARCHAR(255) NULL DEFAULT NULL,
  `role` VARCHAR(255) NULL DEFAULT NULL,
  `username` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `UK_ob8kqyqqgmefl0aco34akdtpe` (`email` ASC) VISIBLE,
  UNIQUE INDEX `UK_sb8bbouer5wak8vyiiy4pf2bx` (`username` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `a2_sd`.`bank_account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`bank_account` (
  `id` INT(11) NOT NULL,
  `creation_date` VARCHAR(255) NOT NULL,
```

```
  `currency` VARCHAR(255) NOT NULL,
  `iban` VARCHAR(255) NOT NULL,
  `last_modified` VARCHAR(255) NOT NULL,
  `sold` DOUBLE NULL DEFAULT NULL,
  `type` VARCHAR(255) NOT NULL,
  `user_id` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `UK_699j998jxie2f134gfnu86q96` (`iban` ASC) VISIBLE,
  INDEX `FK92iik4jwhk7q385jubl2bc2mm` (`user_id` ASC) VISIBLE,
  CONSTRAINT `FK92iik4jwhk7q385jubl2bc2mm`
    FOREIGN KEY (`user_id`)
    REFERENCES `a2_sd`.`user` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `a2_sd`.`bill`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`bill` (
  `id` VARCHAR(255) NOT NULL,
  `accepted_currency` VARCHAR(255) NOT NULL,
  `description` VARCHAR(255) NULL DEFAULT NULL,
  `recipient` VARCHAR(255) NOT NULL,
  `value` DOUBLE NOT NULL,
  `user_id` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FKqhq5aolak9ku5x5mx11cpjad9` (`user_id` ASC) VISIBLE,
```

```
  CONSTRAINT `FKqhq5aolak9ku5x5mx11cpjad9`
    FOREIGN KEY (`user_id`)
    REFERENCES `a2_sd`.`user` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `a2_sd`.`company`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`company` (
  `id` VARCHAR(255) NOT NULL,
  `accepted_currency` VARCHAR(255) NULL DEFAULT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `a2_sd`.`hibernate_sequence`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`hibernate_sequence` (
  `next_val` BIGINT(20) NULL DEFAULT NULL)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;
```

```
-- -----------------------------------------------------
-- Table `a2_sd`.`role`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`role` (
  `id` BIGINT(20) NOT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


-- -----------------------------------------------------
-- Table `a2_sd`.`transaction`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`transaction` (
  `id` VARCHAR(255) NOT NULL,
  `date` VARCHAR(255) NOT NULL,
  `recipient` VARCHAR(255) NULL DEFAULT NULL,
  `type` VARCHAR(255) NOT NULL,
  `value` DOUBLE NOT NULL,
  `bank_account_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FKec44dj1u86xnsku7ld84pirje` (`bank_account_id` ASC) VISIBLE,
  CONSTRAINT `FKec44dj1u86xnsku7ld84pirje`
    FOREIGN KEY (`bank_account_id`)
    REFERENCES `a2_sd`.`bank_account` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;
```

```
-- -----------------------------------------------------
-- Table `a2_sd`.`users_roles`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `a2_sd`.`users_roles` (
  `user_id` VARCHAR(255) NOT NULL,
  `role_id` BIGINT(20) NOT NULL,
  INDEX `FKt4v0rrweyk393bdgt107vdx0x` (`role_id` ASC) VISIBLE,
  INDEX `FKgd3iendaoyh04b95ykqise6qh` (`user_id` ASC) VISIBLE,
  CONSTRAINT `FKgd3iendaoyh04b95ykqise6qh`
    FOREIGN KEY (`user_id`)
    REFERENCES `a2_sd`.`user` (`id`),
  CONSTRAINT `FKt4v0rrweyk393bdgt107vdx0x`
    FOREIGN KEY (`role_id`)
    REFERENCES `a2_sd`.`role` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

b) Inserting into a table:

```
Hibernate:
    update
        hibernate_sequence
    set
        next_val= ?
    where
        next_val=?
Hibernate:
    /* insert ro.sd.a2.entity.BankAccount
        */ insert
        into
            bank_account
            (creation_date, currency, iban, last_modified, sold, type, user_id, id)
        values
            (?, ?, ?, ?, ?, ?, ?, ?)
```

- We have the update as well generated for the auto-incremented id

c) Deleting from a table:

```
Hibernate:
    /* delete ro.sd.a2.entity.Bill */ delete
        from
            bill
        where
            id=?
```

d) Updating a table entry:

```
Hibernate:
    /* update
        ro.sd.a2.entity.BankAccount */ update
            bank_account
        set
            creation_date=?,
            currency=?,
            iban=?,
            last_modified=?,
            sold=?,
            type=?,
            user_id=?
        where
            id=?
```