Requirements to develop the Java application “Bank Client”

**1. Project overview:**

The project is aimed at creating a "Bank Client" application that will allow users to manage their accounts, view the balance and statement for a certain period, as well as make money transfers between user accounts.

**2. Functional requirements:**

**2.1 Users:**

The application must provide the ability to register new users.

Each user must have a unique identifier, name and other necessary data.

**2.2. Balance:**

Each user must have an account.

The account has a balance and a currency that can change as a result of transactions.

**2.3. View balance and statements:**

The user must be able to view their account balance.

The user can view a statement of operations for a certain period (for example, for a day, a week, a month).

**2.4. Money transfer:**

The user must be able to transfer money to another account with automatic conversion to the base currency of the correspondent's account.

Money can be transferred between own accounts and accounts of different users.

**3. Technical requirements:**

**3.1. Technologies:**

* The application must be implemented in the Java 8-21 programming language.
* Building the project using maven
* Spring Boot
* PostgreSQL.

**3.2. Data storage:**

Data about users and their accounts must be stored:

* in the file. (milestone 1)
* in the database. (milestone 4) It is recommended to use a relational database such as PostgreSQL.

**3.3. Interaction with the user:**

* no interface (milestones 1-3)
* restless API (milestone 3)

**3.4. Data protection:**

Ensure the end user data privacy.

Use encryption to store sensitive data such as passwords.

**3.5. Error handling:**

Handle possible data entry errors.

Provide debugging and error logging for further analysis and correction.

**4. Additional requirements:**

**4.1. Transaction logging:**

Provide a log of transactions for each account.

The log should contain transaction details such as date, time, amount, type of transaction (transfer, top-up, etc.).

**5. Testing tasks:**

**5.1. Unit tests (milestone 2):**

Write unit tests to check the operability of individual application components (for example, classes, methods).

**5.2. Integration tests (Optional milestone 4):**

Write integration tests to test the interaction between different components of the application (for example, the database).

**Milestone 1 (2 months)**

**Simple Java** [**project**](http://dolszewski.com/java/project-package-organization/) **that will be executed in main**

And will have the following possibilities:

* A method that adds a user.
* A method that adds an account.
* A method that transfers funds from one account to another.
* A method that outputs a list of users to the console.
* A method that outputs the data of a specific user to the console.
* A method that displays the accounts of a specific user in the console.
* A method that displays an account statement for a certain period.
* All these methods should be covered by unit tests.

After each change, the data must be saved in a file. When opening the application, data about transactions, accounts, and users must be read from the file.

**Main Goals:**

1. Java Syntax
2. Comparison of objects in java. Like: !=, ==, >, <, >=, <=, equals(), hashCode(), comparators.
3. Collections knowledge (Medium knowledge required: must know which collection is sortable and which is not, and which collection is used for which purposes)
4. Exceptions (What kind of exceptions are used in java. Exception hierarchy)
5. Practical application of the principles of OOP. Inheritance, polymorphism and encapsulation
6. Input/Output stream – it is commonly used in Java. It can be called an interface between the foreign world and Java application.

# **Milestone 2 (1.5 months)**

**Maven (common libs: guava та apache-commons) + Spring Boot + Lombok +Log4j**

**Integration of** [**maven**](https://spring.io/guides/gs/maven) **and** [**Spring boot**](https://spring.io/guides/gs/spring-boot) **into the project**

The project must be built using maven (unit tests from the previous milestone must be run using maven) and [should be executed as:](https://docs.spring.io/spring-boot/docs/current/reference/html/executable-jar.html)

***java -jar bank\_app.jar***

all services/managers should be Spring [beans](https://docs.spring.io/spring-framework/reference/core/beans/definition.html)

**Keywords**:

* Service
* Bean
* Autowired
* Log4j2
* AllArgsConstructor

**Main Goals:**

1. Knowledge of how to compile, build, and run a Java project with Maven.
2. Spring Core essentials, such as the dependency injection container, bean scopes, and common annotations for these purposes.
3. Basic knowledge of how to use a logging framework and its purposes.

# **Milestone 3 (4 weeks)**

**Rest API + integration test**

All methods specified in point 1 must have a corresponding [endpoint](https://spring.io/guides/gs/rest-service) and (GET/POST….) and have validation and covered by integration(optional) tests.

**Keywords:**

* RequestBody
* RequstParam
* RestController
* ExceptionHandler
* PostMapping
* RequestMapping
* ResponseBody
* PathVariable
* Validated
* Valid

**First of all, I would like to ask you to draw the API in the scheme and approve it with the mentor.**

**Main Goals:**

1. Knowledge of how to work with the HTTP protocol using Spring and the default web server inside the framework.
2. Use of integration tests and understanding the difference between unit and integration tests.

# **Milestone 4 (1 month)**

**Database + Liquibase + integration tests(optional)**

At this stage, a database should be integrated instead of a file. The scheme should roll out using Liquibase with initial data.

It is also necessary to add transactionality to the corresponding methods.

**First of all, I would like to ask you to draw up the DB structure and approve it with the mentor.**

**Keywords**:

* spring-boot-starter-data-jpa
* jdbc
* JDBCTemplate
* RowMapper
* hikari
* Transactional
* Repository

**Main Goals:**

1. Basic knowledge of table relationships.
2. Modeling tables according to business requirements.
3. Using SQL queries to integrate a Java application with a database.
4. Use of integration tests and understanding the difference between unit and integration tests.

# **Milestone 5 (3 weeks)**

**Quartz + Cache + HTTP calls**

A job should be implemented that requests currency exchange rates via [API](https://freecurrencyapi.com/docs/#official-libraries) and adds them to the cache. When transferring money from accounts with different currencies this cache must be used.

**Keywords**:

* Cacheable
* Quartz
* RestClient
* RestTemplate

**Main Goals:**

1. When and how cache can be used.
2. HTTP calls in java.
3. Basic knowledge about Quartz jobs.