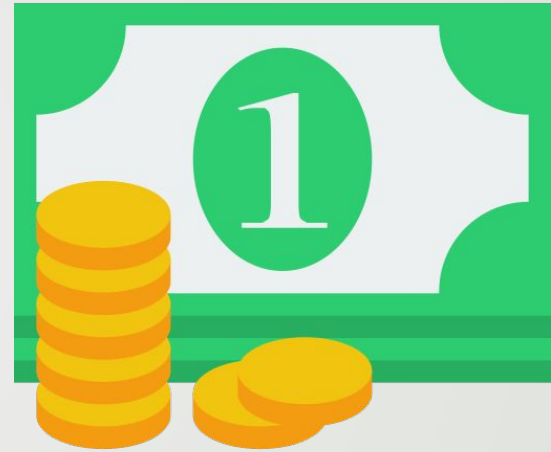


# eBank web app

Development  
process



\*All usernames, passwords or other information shown in this presentation is fictive and for illustrative purpose only

# TABLE OF CONTENTS

---

**01**

---

Implementing the web app initial structure

---

**02**

---

Displaying banking transaction history

---

**03**

---

Displaying total balance

---

**04**

---

Displaying total money in, total money out, and interest

---

**05**

---

Implementing the login logic

---

**06**

---

Implementing money transfer between users

# TABLE OF CONTENTS

---

**07**

---

Implementing the loan feature

---

**08**

---

Implementing the delete account box

---

**09**

---

Implementing the sorting option

---

**10**

---

Formatting dates and currencies

---

**11**

---

Implementing security automatic logout

---

**12**

---

Finalizing code refactoring and running final tests

# TABLE OF CONTENTS

---

**13**

---

Deploying the web app online

---

**14**

---

Completing the README document



**Implementing** the web app  
initial structure

---

Skills used

---

N.O.

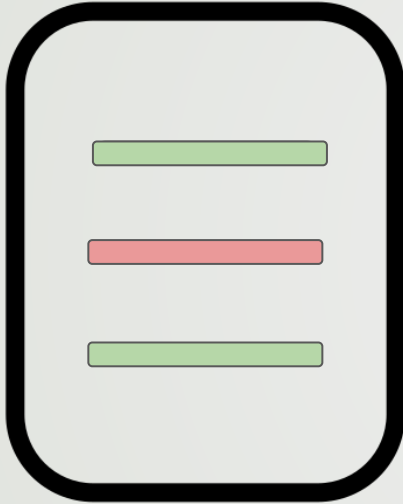
# WHY WAS THIS STEP IMPORTANT

\*HTML and CSS templates were made available / created by Jonas Schmedtmann.

This step was important to ensure that the initial structure of the web app was properly established (structure on which JavaScript would then be able to work with).

# WHAT WAS THE GOAL

Ensuring that all static content for the web app is present in the HTML and CSS documents, with appropriate class names and ids to allow easy reference to it later using JavaScript.



## Displaying banking transaction history

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

This step was important to allow users to see each transaction carried out in their account, and thus have a good overview of the financial activities carried out.

## WHAT WAS THE GOAL

- Using mock user accounts and mock data, the objective was to display on the screen each operation carried out by a user.
  - To do this, a logic to identify whether each operation is a money inflow (deposit) or outflow (withdrawal) was created, and stored in a variable (*type*).
- A template literal was then created as well, using the variable previously defined (*type*) to dynamically select the right HTML class in the DOM, and the right CSS element associated with it to make the deposits appear green, and the withdrawal red.

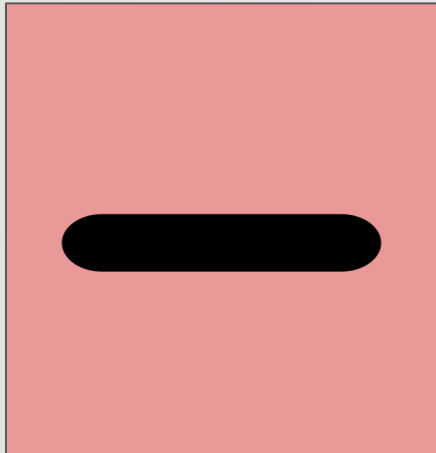
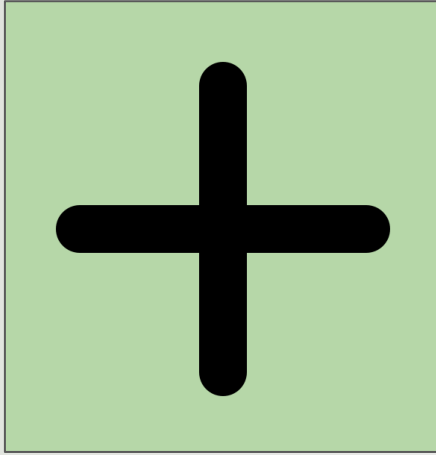


# FIRST VIEW OF THE **OPERATIONS HISTORY**

Deposit →

€ DEPOSIT	1300
€ DEPOSIT	70
€ WITHDRAWAL	-130
€ WITHDRAWAL	-650
€ DEPOSIT	3000
€ WITHDRAWAL	-400

Withdrawal →



---

03

---

**Displaying** total balance

---

Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

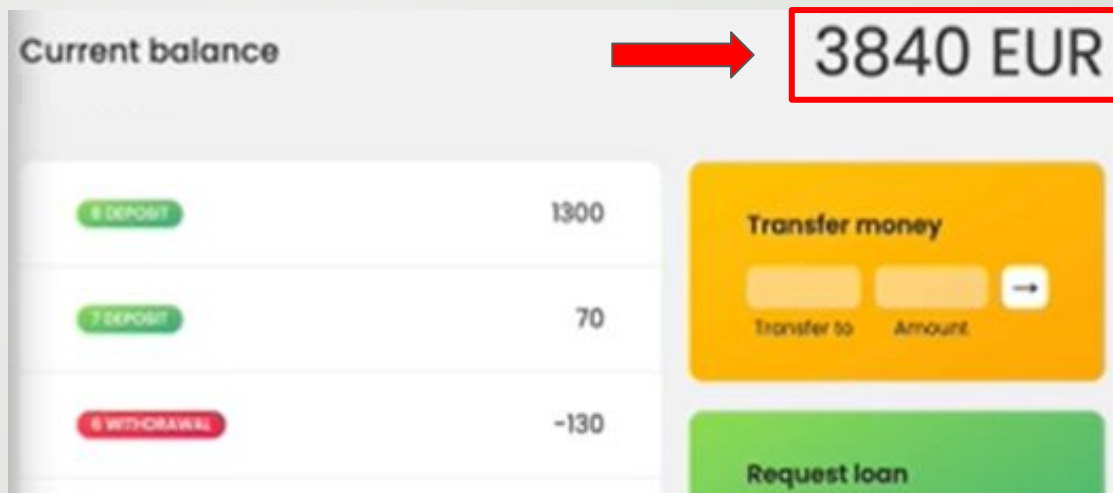
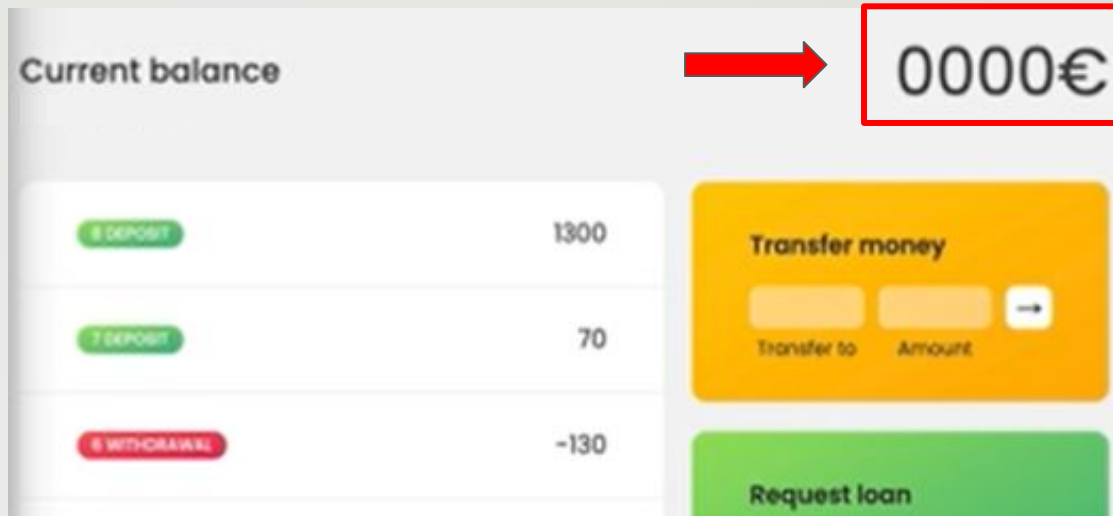
The total balance of the bank account is one of the main information displayed in any banking app, so it was important to calculate it and ensure that it is displayed correctly.

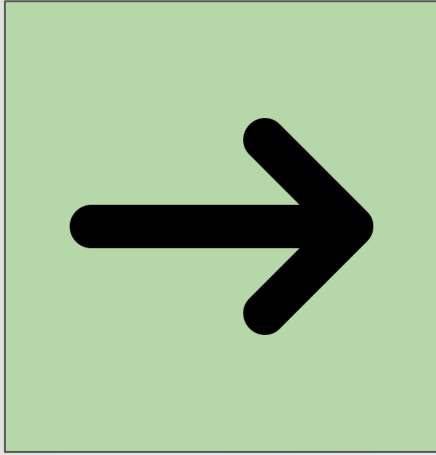
## WHAT WAS THE GOAL

Calculating the balance of user's bank account based on their banking history (deposits and withdrawals), then display this balance.

- To do so, the *reduce()* method have been used to process each user banking operation array (array that contains all amounts of deposits and withdrawals for a user), in order to return a single balance value.
- For example, a user having 2 deposits of 100\$ and 1 withdrawal of 25\$ would have 175\$ as a balance.

# BALANCE CALCULATED AND DISPLAYED



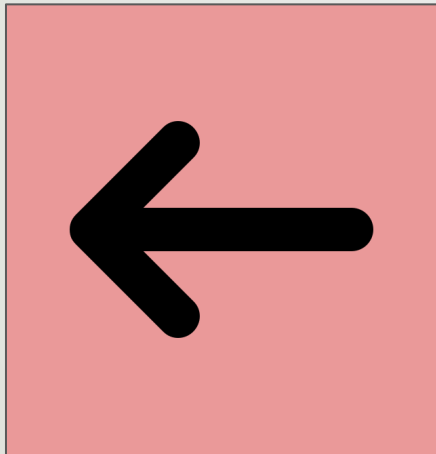


---

04

---

**Displaying** total money in,  
total money out, and interest



---

Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

This step was important to present in a simple and quick way a summary of users' money inflows and outflows.

# WHAT WAS THE GOAL

From the users' transaction history, separating the deposits from the withdrawals (using the *filter()* method), and then add respectively all the deposits together, and all the withdrawals together, to obtain a single value for each (by chaining *reduce()* method to the *filter()* method).

Interests were also calculated and displayed at this point.

# FIRST VIEW

## SUMMARY

DEPOSIT	3000
WITHDRAWAL	-400
DEPOSIT	450
DEPOSIT	200

Request loan

Amount

Close account

user

IN 0000€ OUT 0000€ INTEREST 0000€

↓ SORT

You will be logged out in 05:00

# UPDATED VIEW

## SUMMARY

DEPOSIT	3000€
WITHDRAWAL	-400€
DEPOSIT	450€
DEPOSIT	200€

### Request loan

Amount

### Close account

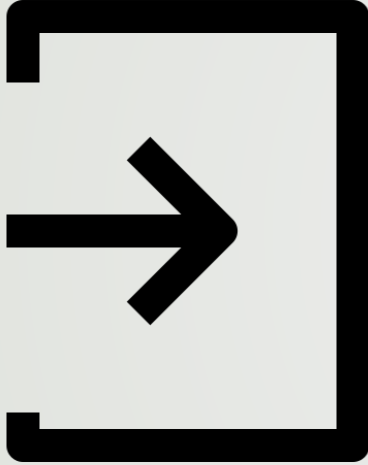
Confirm user Confirm Pin

in 5020€ out 1180€ interest 59.4€

↓ SORT

You will be logged out in 05:00





## **Implementing** the login logic

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

This step was important to ensure that each user has a personalized experience, and that the financial information displayed is unique and specific to each user.

# WHAT WAS THE GOAL

Implementing a simple system allowing users to log in with a username and a PIN.

\*However, since the focus of this project was primarily on working with arrays, DOM manipulation and functions, the login system has not been developed as much as it could have been, on purpose. Since users information is also not stored in a database, but rather hard coded in the JS file for testing purpose, the login system remains quite basic, and several other features could have been added for a more complete and realistic log in process.

# LOGIN PAGE

Log in to get started

eB

user

PIN

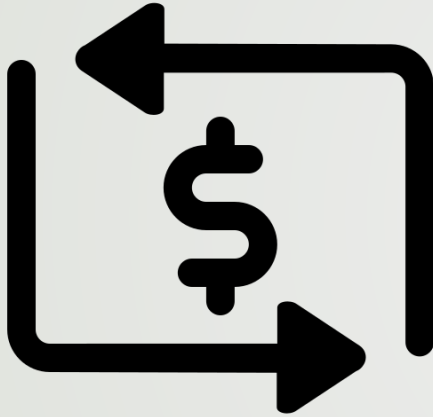


user

PIN



SIMPLE SYSTEM LOOKING IF USERNAME AND PIN MATCH  
CORRESPONDING DATA IN THE USER OBJECT



## Implementing money transfer between users

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

Transferring money between people is one of the common operations. It was therefore important to allow users to do the same with eBank web app.

## WHAT WAS THE GOAL

Ensuring that users can transfer money to each other. To do this, two main parts were worked on:

- A part allowing users to identify the receiver account, and to transfer the money there correctly.
  - The *find()* method was used for this.
- A part allowing users to define the amount to be transferred.

\*In both parts, several checks and validations have been coded to prevent users from doing incorrect operations (ex: sending more money than they have, or sending money to a receiver account that does not exist).

Updating in real time, after each transfer, user's UI (balance, operation history, and summary) for both the sender and the receiver.

# MONEY TRANSFER BOX

Welcome back Jake

eB

user

PIN



Current balance

25 952,59 €

8 DEPOSIT

1300,00 €

7 DEPOSIT

79,97 €

6 WITHDRAWAL

-133,90 €

5 WITHDRAWAL

-642,21 €

4 DEPOSIT

25 000,00 €

Transfer money

Transfer to

Amount



Request loan

Amount



# TRANSFERRING MONEY TO ANOTHER USER

Welcome back Jake

eB

user PIN →

Current balance

25 952,59 €

8 DEPOSIT

1300,00 €

7 DEPOSIT

79,97 €

6 WITHDRAWAL

-133,90 €

5 WITHDRAWAL

-642,21 €

4 DEPOSIT

25 000,00 €

Transfer money

ew

1000

→

Transfer to Amount

Request loan

→

Amount

# TRANSFERRING MONEY TO ANOTHER USER

(SAME UPDATE IN THE **RECEIVER ACCOUNT** TO SHOW THE DEPOSIT TRANSFER +1000)

Welcome back Jake

eB

user PIN →

Current balance

24 952,59 €

9 WITHDRAWAL -1000,00 €

8 DEPOSIT 1300,00 €

7 DEPOSIT 79,97 €

6 WITHDRAWAL -133,90 €

5 WITHDRAWAL -642,21 €

Transfer money

Transfer to Amount →

Request loan

Amount →

Transaction	Amount
9 WITHDRAWAL	-1000,00 €
8 DEPOSIT	1300,00 €
7 DEPOSIT	79,97 €
6 WITHDRAWAL	-133,90 €
5 WITHDRAWAL	-642,21 €





## Implementing the loan feature

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

Requesting a loan from a bank online is a very common operation for people. It was therefore important to allow users to do the same with eBank web app.

## WHAT WAS THE GOAL

Ensuring that users can request a loan. To do this, two main parts were worked on:

- A part allowing users to define the requested loan amount.
  - Two checks/validations have been added to ensure that the users request an amount greater than 0, and that users have at least 1 deposit greater or equal to 10% of the requested loan.
- A part in which a timer has been implemented, simulating the approval time required for the loan (delay of three seconds).

Updating user's UI automatically after loan approval with the new correct numbers (balance, operation history, and summary).

# LOAN REQUEST BOX

Welcome back Jake



user

PIN



Current balance

25 952,59 €

8 DEPOSIT

1300,00 €

7 DEPOSIT

79,97 €

6 WITHDRAWAL

-133,90 €

5 WITHDRAWAL

-642,21 €

4 DEPOSIT

25 000,00 €

Transfer money

Transfer to

Amount



Request loan

Amount



# REQUESTING A LOAN

Welcome back Jake



user

PIN



Current balance

25 952,59 €

8 DEPOSIT

1300,00 €

7 DEPOSIT

79,97 €

6 WITHDRAWAL

-133,90 €

5 WITHDRAWAL

-642,21 €

4 DEPOSIT

25 000,00 €

Transfer money

Transfer to

Amount



Request loan

3000

Amount



# REQUESTING A LOAN

Welcome back Jake



user

PIN



Current balance

28 952,59 €



9 DEPOSIT	3000,00 €
8 DEPOSIT	1300,00 €
7 DEPOSIT	79,97 €
6 WITHDRAWAL	-133,90 €
5 WITHDRAWAL	-642,21 €

Transfer money

Transfer to

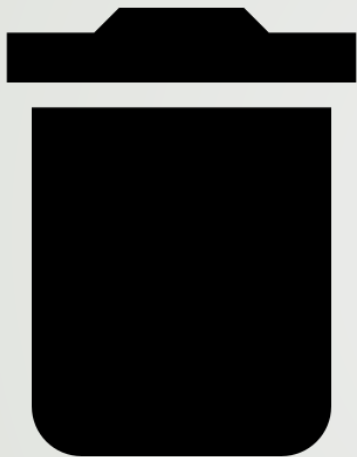
Amount



Request loan

Amount





## Implementing the delete account box

---

Skills used

---

Code writing

# WHY WAS THIS STEP IMPORTANT

Allowing users to close their accounts is a basic operation that users generally expect to have. It was therefore important to allow users to do the same with eBank web app.

# WHAT WAS THE GOAL

Ensuring that users can delete their account by using their username and PIN.

- The *findIndex()* and *splice()* methods have been used for this.

# CLOSE ACCOUNT BOX

The image shows a mobile application interface. On the left is a list of transactions, and on the right is a sidebar menu. The sidebar menu has four items: 'Transfer money' (yellow), 'Request loan' (green), 'Close account' (pink, highlighted with a red box and a red arrow), and 'Close account' (pink, highlighted with a red box and a red arrow). The 'Close account' item is the one being pointed to by the red arrow.

Transaction ID	Type	Amount
8	DEPOSIT	1300,00 €
7	DEPOSIT	79,97 €
6	WITHDRAWAL	-133,90 €
5	WITHDRAWAL	-642,21 €
4	DEPOSIT	25 000,00 €
3	WITHDRAWAL	-306,50 €
2	DEPOSIT	455,23 €

### Transfer money

Transfer to Amount

### Request loan

Amount

### Close account

Confirm user Confirm PIN





## Implementing the sorting option

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# **WHY** WAS THIS STEP IMPORTANT

Sorting financial operations history in an orderly way helps to have a better overview, and improves user experience.

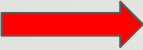
# **WHAT** WAS THE GOAL

Implementing a feature (using the *sort()* method) to allow users to display their financial transactions from the largest deposit to the largest withdrawal, and to return to a chronological display when desired.

# CHRONOLOGICAL DISPLAY

Current balance

25 952,59 €



8 DEPOSIT	1300,00 €
7 DEPOSIT	79,97 €
6 WITHDRAWAL	-133,90 €
5 WITHDRAWAL	-642,21 €
4 DEPOSIT	25 000,00 €
3 WITHDRAWAL	-306,50 €
2 DEPOSIT	455,23 €

Transfer money

→

Transfer to Amount

Request loan

→

Amount

Close account

→

Confirm user Confirm PIN


IN 27 035,20 € OUT 1082,61 € INTEREST 323,46 € ↓ SORT

You will be logged out in 04:52

# SORTED DISPLAY

Current balance

25 952,59 €



8 DEPOSIT	25 000,00 €
7 DEPOSIT	1300,00 €
6 DEPOSIT	455,23 €
5 DEPOSIT	200,00 €
4 DEPOSIT	79,97 €
3 WITHDRAWAL	-133,90 €
2 WITHDRAWAL	-306.50 €

Transfer money

Transfer to Amount

Request loan

Amount

Close account

Confirm user Confirm PIN

IN 27 035,20 € OUT 1082,61 € INTEREST 323,46 € [SORT](#)



You will be logged out in 04:38



## Formatting dates and currencies

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

Considering that users may come from multiple regions around the world, it was important to adjust automatically the date format and the currency displayed accordingly, to avoid any possible confusions among users.

# WHAT WAS THE GOAL

Displaying the correct date format and currency in each user's UI regardless of their location in the world / ensure users don't have to configure themselves these parameters.

- To do so, the *Internationalization* API provided by JavaScript, and more precisely the *Intl.DateTimeFormat* and the *Intl.NumberFormat*, have been used.

# DATE AND CURRENCY EUROPE FORMAT

Current balance

As of 07/02/2024, 12:49

25 952,59 €

8 DEPOSIT

YESTERDAY

1300,00 €

7 DEPOSIT

2 DAYS AGO

79,97 €

6 WITHDRAWAL

6 DAYS AGO

-133,90 €

5 WITHDRAWAL

08/05/2020

-642,21 €

4 DEPOSIT

01/04/2020

25 000,00 €

Transfer money

Transfer to

Amount



Request loan

Amount



# DATE AND CURRENCY US FORMAT

Current balance

As of 2/7/2024, 12:55 PM

\$11,720.00

8 WITHDRAWAL

7/26/2020

-\$30.00

7 DEPOSIT

6/25/2020

\$8,500.00

6 WITHDRAWAL

4/10/2020

-\$1,000.00

5 WITHDRAWAL

2/5/2020

-\$3,210.00

4 WITHDRAWAL

1/25/2020

-\$790.00

Transfer money

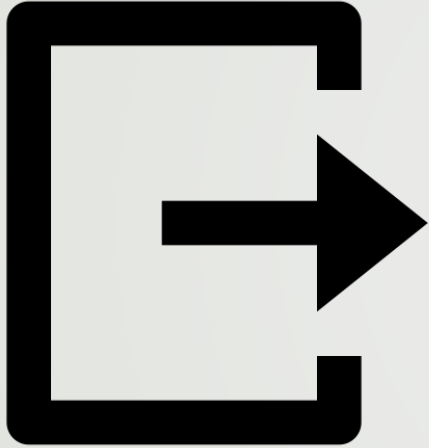
Transfer to

Amount

Request loan

Amount





## **Implementing** security automatic logout

---

### Skills used

---

Research  
Problem solving  
Code writing  
Debugging

# WHY WAS THIS STEP IMPORTANT

User security is always very important in a banking application context. Implementing this feature was therefore necessary.

## WHAT WAS THE GOAL

Creating a timer to track each user activity. If a user does not perform any operations within 5 minutes, their account is automatically logged out for security reasons.

- For each operation carried out by users (money transfer, loan request or sorting financial transactions), the timer is automatically set back to its starting time and re-begins to count down, thus avoiding unwanted logout.
- To do this, a function (*startLogoutTimer*) was created to display the timer on users' UI and track inactivity time.

# Current balance

As of 07/02/2024, 13:11

25 952,59 €

8 DEPOSIT	YESTERDAY	1300,00 €
7 DEPOSIT	2 DAYS AGO	79,97 €
6 WITHDRAWAL	6 DAYS AGO	-133,90 €
5 WITHDRAWAL	08/05/2020	-642,21 €
4 DEPOSIT	01/04/2020	25 000,00 €
3 WITHDRAWAL	28/01/2020	-306,50 €
2 DEPOSIT	23/12/2019	455,23 €

## Transfer money

→

Transfer to

Amount

## Request loan

→

Amount

## Close account

→

Confirm user

Confirm PIN

IN 27 035,20 € OUT 1082,61 € INTEREST 323,46 € ↓ SORT



You will be logged out in 04:55



**Finalizing** code refactoring  
and **running** final tests

---

Skills used

---

Code writing

# **WHY** WAS THIS STEP IMPORTANT

When possible, it is good practice to refactor codes. When there are a lot of duplicate codes, and some functionalities need to be changed for example, multiple identical updates may be necessary across various locations and files, leading to a potentially lengthy and error-prone process. Code refactoring helps prevent this by making the code cleaner, more logical, and more concise.

# WHAT WAS THE GOAL

Replacing some previously duplicated lines of code to accomplish the same actions, but with fewer code lines, thus significantly reducing the length of the code. To do this, different functions were created.

Testing the web app with all possible scenarios that users might encounter to ensure everything works as expected.

Fixing bugs as needed.

Adding comments where necessary.



github-pages

**Deploying** the web app online

---

Skills used

---

N.O.

# **WHY** WAS THIS STEP IMPORTANT

This step was important to make the eBank web app publicly available by hosting it on GitHub Pages (gh-pages).





README.



md

## Completing the README document

---

### Skills used

---

Communication  
Content writing

# WHY WAS THIS STEP IMPORTANT

Ensure the project is well documented and easily accessible by anyone interested.

# WHAT WAS THE GOAL

Updating and completing the README file located in the *eBank-web-app* Github repository. The goal was to ensure that all relevant information regarding this project is accessible under these three categories:

- Project description
- User interface
- Technical aspects

# README SAMPLE - FULL VERSION ON GITHUB

