**ATM Application**

**-**

**Senila Constantin**

**Faculty of Automations and Computer Science**

**Computer Science Department - 4th year**

* YouTube link for the video of the application**:** [**https://www.youtube.com/watch?v=ruL7wwfdesk**](https://www.youtube.com/watch?v=ruL7wwfdesk)
* GitHub code link: <https://github.com/senilaconstantin/ATM_application.git>

## Critical & Logic rationale

Answer: icon A

## Among icons A, B and C, icon A should be chosen, because when icon A was placed in the first position, it obtained a higher result than icon C, also in the same position. Considering that icon B was not on the first position, it was only on position 2 and 3, we can compare with icon A for position two where icon A has a slightly higher score. We can also compare icon B with icon C on position 3 where icon B has a higher score than icon C. For these reasons I would choose icon A because it obtained a higher score on the positions it was on than the other two icons in the same positions. Making a ranking, the first place would be the A icon, the second place would be the B icon, and the 3rd place would be the C icon.

## Technologies

What would you use to implement an API call and why?

To implement an API call I used the Alamofire library in Swift. Alamofire is a very popular Swift library used to make HTTP API calls.

Some of the reasons I used Alamofire include:

* Simplified Requests: Alamofire provides a simple and easy-to-understand syntax for creating and sending HTTP requests. This can make the development process much faster and more efficient.
* Response handling: Alamofire provides a number of options for handling API responses, including decoding JSON responses into Swift objects. This can make the data parsing process much easier and faster.
* Error Handling: Alamofire comes with a number of features for error handling, including network error handling, API response error decoding, and JSON parsing error handling. This can make the error handling process much simpler and more efficient.

How to store data locally in an iOS app?

For saving data locally in the ATM application that I made in iOS, I used UserDefaults. To save the data a Userdefaults I used '.set(encodedData, forKey: "card")' which saves the carded object to the "card" key in UserDefaults. Before saving the object, it is transformed into JSON format by means of an encoder. Using the method '.object(forKey: "card")' I take the object from the key "card", and then I decode it using JSONDecoder. To delete the card data I set the object attributes as empty.

## Object Oriented Programming

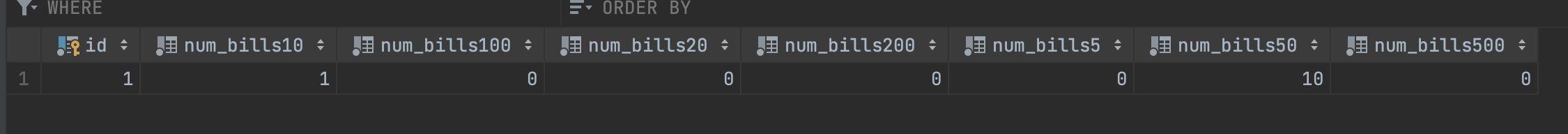
1. **Application**

* YouTube link for the video of the application**:** [**https://www.youtube.com/watch?v=ruL7wwfdesk**](https://www.youtube.com/watch?v=ruL7wwfdesk)
* GitHub code link: <https://github.com/senilaconstantin/ATM_application.git>

In this application we used iOS (Swift), Java and PostgreSql for implementation.

I will present some cases below:

1. If a transaction is attempted, but there are not enough banknotes for the respective amount:



A screenshot of a message

Description automatically generated with medium confidence

1. If there is not enough money in the account:

A picture containing text, screenshot, font, number

Description automatically generated

A screenshot of a message

Description automatically generated with medium confidence

1. If the sum is not a multiple of 10:

A screenshot of a message

Description automatically generated with medium confidence

1. If no problem was encountered:

A screenshot of a yellow box with blue buttons

Description automatically generated with low confidence

After each operation, this screen appears, and if the yes button is pressed, the user is redirected to the main page, and otherwise to login (the card is released).

**API:**

1. Add Card:

**A screenshot of a computer

Description automatically generated with medium confidence**

1. LogIn Card:

**A screenshot of a computer program

Description automatically generated with medium confidence**

1. Withdraw:

A screenshot of a computer

Description automatically generated with medium confidence