**Web Services and API Development: Final Project**

Team G:

Gheorghe Strugaru 16112385

Imran Zulfiqar 16112369

Witold Zolnowski 17143853

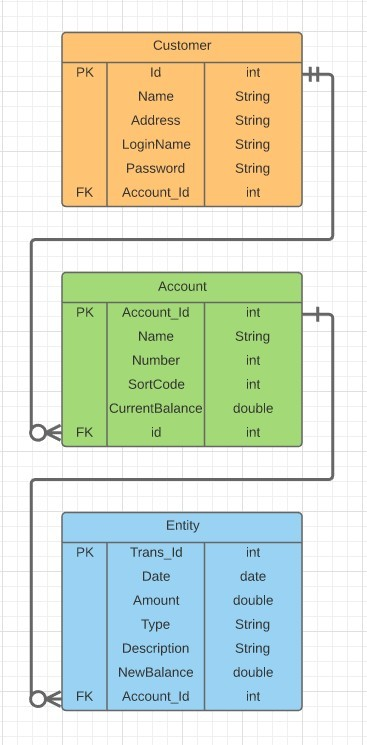
1. Problem description and proposed solution:

The task is to design API services. Problem: ensure that the project structure will facilitate security while simultaneously being intuitive, and easy to manage. One of the main problems initially was how should the api structure look like. In order to solve this problem we decided to rely on the restful design of services, and at the same time, to ensure that there is a clear separation of duties. To improve future debugging and overall understandability of the code base, we have followed the basic design patterns, and split the project asssets into three separate groups: models, resources, and services. Furthermore, we have decided to use version control available via github to ensure that we have reliable backups. Finally, we developed the project incrementally, aiming initially to build MVP (minimum viable product). Once achieved, we have proceeded to expand services, and add functionalities. Overall, we found the project challenging in a positive way, as it pushed us out of our comfort zones, and forced to think through design patterns, and how to structure the overall project.

2. Security concerns

3. Entity-Relationship diagram

There are three entities: Customer, Account, and Transaction. Customer and Account have one, and only one to many relationship. Account and Transaction have one to many relationship.



**API entry points:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| DEFAULT | API entry point | /banking | |  |  |  |  |
| Customer Resources | Resources related to customer records | /banking /customers | |  |  |  |  |
| getAllCustomers | Retreives all customers | / | GET | no params | Returns List of all customer objects | No preconditions. If no customers, returns empty array | No changes to the system state, returns array. |
| getCustomer | Retreives single customer | /{id} | GET | int id | Returns Customer object | Customer must existst in system | No changes to the system, object returned |
| createCustomer | Creates new customer record | / | POST | Customer object | Returns customer object as confirmation | Customer must not already exist in database | Customer record created, database updated |
| updateCustomer | Updates customer record | /{id} | PUT | int id,  Customer object | Returns updated customer object | Customer must exist in the database | Customer record updated in database |
| deleteCustomer | Removes customer record | /{id} | DELETE | int id | Returns customer object that has been removed | Customer must exist in the database | Customer record gets removed permanently from database |
| Account Resources | Resources related to accounts operations | /banking/customers/{id}/accounts | |  |  |  |  |
| createAccount | Creates a new account for customer | / | POST | int id,  String type (account name) | Returns newly created account object | Customer must exist, customer must be logged in | New account is added to customer  's account array |
| getAllAccounts | returns all accounts that belong to spcified customer | / | GET | int id | Returns array list of accounts | Customer must be logged in. If no accounts, returns empty array | No effect on system, returns array |
| getAccount | Looks for and returns specific account | /{accountNumber} | GET | int id,  int accountNumber | Returns account object | Customer must be logged in, account must exist for logged in user | No effect on system, returns account object. |
| getBalance | Looks at the balance of specified account | /{accountNumber}/balance | GET | int id,  int accountNumber | Returns string containing balance of the account | Customer must be logged in, account must exist. | No effect on system, displays balance on client's app. |
| deleteAccount | Removes account from database | / | DELETE | int id, Account a | Returns string confirming removal of account | Customer must be logged in, account must exist | Removes permanently account for given customer with all related information |
| Transaction Resources | Resources related to account transactions | /banking/customers/{id}/accounts/{accountNumber}/transactions | |  |  |  |  |
| getTransaction | Retreives single transaction that maches specified amount | /{amount} | GET | int id,  int accountNumber,  double amount | Returns transaction object | Customer must be logged in, there has to be amount specified, queried account has to exist | Returns and displays transaction details on client side |
| getAllTransactions | Retreives all transactions for specific account | / | GET | int id,  int account id | Returns array of transaction objects | Customer must be logged in, account must exist | Displays list of transactions on client's side. |
| withdrawFromAccount | Withdraws funds from specified account, creates related transaction | /withdraw/{amount} | GET | int id,  int accountNumber,  int amount | Returns transaction object | Customer must be logged in, account must exist, there must be sufficient funds on account | Displays message on clients side if transaction successful, creates and records transaction for affected account. Updated account current balance. |
| lodgeToAccount | Lodges specified amount to specified account | /lodge/{amount} | GET | int id,  int accountNumber,  double amount | Returns transaction object | Customer must be logged in, account must exist, account must belong to customer | Displays message on clients side if transaction successful. Creates and records transaction to database. Updates affected accounts current balance |
| transferBetweenAccounts | Transfers funds between two accounts | /{amount} | POST | int id, int accountNumber, double amount, Account destinationAccount | Returns transaction object | Customer must be logged in, both accounts must exist, origin (account being credited) account must belong to customer! | Displays message on client's side. Creates two transaction objects, records both objects to relevant acconts. Updates balances of both accounts. |