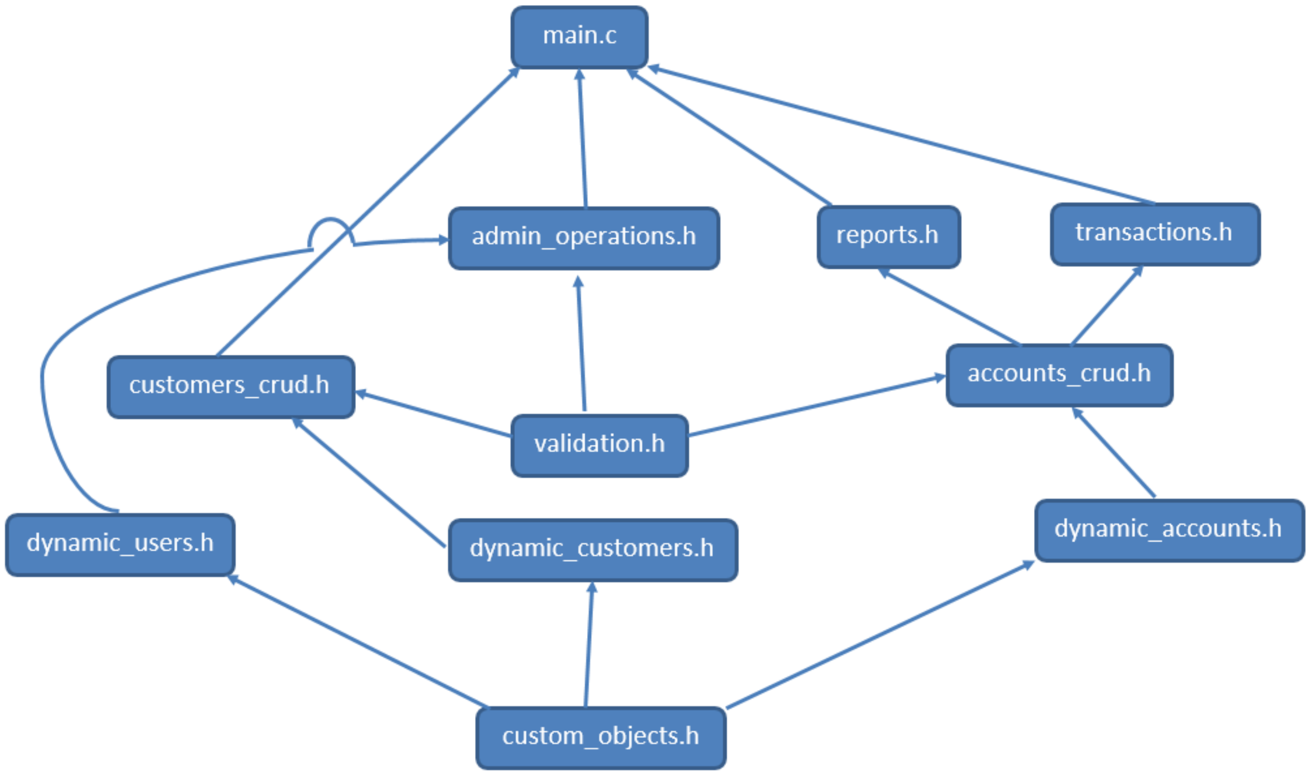
Project 2 Financial Data Management – Documentation

Student: Moroz Alexandra-Ioana

UBB informatică română, anul 1

STRUCTURE OF THE PROJECT:



CUSTOM_OBJECTS.H

Contains custom struct objects for handling data about a client.

struct customer (char name[], char iban[], char phone[], char id_string[], char email[])

struct Node_customer (customer data, struct Node_customer* next)

struct account (char type, float balance, char iban[])

struct Node_account (account data, struct Node_account* next)

struct user(char id[], char username[], char password[])

struct Node_user (user data, struct Node_user* next)

VALIDATION.H

Contains validation methods for frequently used user input.

int validare_string(char x[])

- validates an array of characters, without digits, spaces or commas
- returns 1 for a valid array and 0 otherwise

int validare_iban(char x[])

- validates an iban code that complies with the format "AB49ABCD1B31007593840000":
-     * first two characters -> letters
-     * next two characters -> digits
-     * next four characters -> letters
-     * remaining characters -> alphanumeric characters
-     * total of 24 characters
- returns 1 for a valid array and 0 otherwise

int validare_phone(char x[])

- validates a phone number – array with numeric characters
- returns 1 for a valid array and 0 otherwise

int validare_id(char x[])

- validates an id – array with numeric characters
- returns 1 for a valid array and 0 otherwise

int validare_email(char x[])

- validates an email address – array that contains the symbol „@"
- returns 1 for a valid array and 0 otherwise

int validate_amount(char s[])

- validates an amount (number with two decimal places)
- returns 1 for a valid array and 0 otherwise

int check_username(char user[])

- checks if the string provided corresponds to an unique username
- returns 1 if there isn't any client with the same username and 0 otherwise

int check_user_id(char id[])

- checks if the string provided corresponds to an unique id
- returns 1 if there isn't any client with the same id and 0 otherwise

int validate_password(char pass[])

- checks if the string provided corresponds to a valid password (has at least 4 characters and no spaces)
- returns 1 if the password is valid and 0 otherwise

DYNAMIC_CUSTOMERS.H

Contains basic methods for dynamically allocating the customers list.

Imports header CUSTOM_OBJECTS.H

int check_id_customer(struct Node_customer* head, char new_id[])

- checks the existence of an object with the given id in the list starting at address head
- returns 0 if another object which meets the condition is found and 1 otherwise

struct Node_customer* insert_at_end_customer(struct Node_customer* head, char name[], char iban[], char phone[], char id[], char email[])

- inserts a new customer with given data at the end of the list starting at address head
- returns the current address of the start of the list
- exception: memory allocation error

void modify_by_id_customer(struct Node_customer* head, char id[], char name[], char iban[], char phone[], char email[])

- modifies (in place) a customer with a given id and data

struct Node_customer* delete_by_id_customer(struct Node_customer* head, char id[])

- deletes a customer with a given id and data from the dynamically allocated list
- returns the current address of the start of the list

CUSTOMERS_CRUD.H

Contains CRUD operations on the customers list and file.

Imports headers DYNAMIC_CUSTOMERS.H and VALIDATION.H

void save_customers_to_file(struct Node_customer *head, char global_user[])

- saves all the customers in dynamically allocated list starting at address head to file .../customers.txt
- exception: Error opening file

struct Node_customer* load_customers(struct Node_customer *head, char global_user[])

- loads all the customers in dynamically allocated list starting at address head from file .../customers.txt
- exception: Error opening file
- returns the current address of the start of the list

void print_all_customers(struct Node_customer *head)

- prints all the customers from the dynamically allocated list starting at address head
- exception: No customers available

struct Node_customer* add_customer(struct Node_customer *head, char global_user[])

- adds a new customer to the list starting at address head and into the .../customers.txt file
- exception: Error opening file at path
- returns the current address of the start of the list

void modify_customer(struct Node_customer *head, char global_user[])

- modify a customer identified by id from the list starting at address head and into the .../customers.txt file
- exception: Error opening file at path, Customer not found

struct Node_customer * delete_customer(struct Node_customer *head, char global_user[])

- deletes a customer identified by id from the list starting at head and from the .../customers.txt file
- exception: Error opening file at path, Customer not found
- returns the current address of the start of the list

DYNAMIC_ACCOUNTS.H

Contains basic methods for dynamically allocating the accounts list.

Imports header CUSTOM_OBJECTS.H

int check_id_account(struct Node_account* head, char new_id[])

- checks the existence of an object with the given id in the list starting at address head
- returns 0 if another object which meets the condition is found

- return -1 if the account found has balance zero
- otherwise return 1

struct Node_account* insert_at_end_account(struct Node_account* head, char type[], char iban[], float value)

- inserts a new account with given data at the end of the list starting at address head
- returns the current address of the start of the list
- exception: memory allocation error

void modify_account_by_id(struct Node_account *head, char id_string[], float value)

- modifies (in place) an account with a given id and balance

struct Node_ account * delete_by_id_ account (struct Node_ account * head, char id[])

- deletes an account with a given id and data from the dynamically allocated list
- returns the current address of the start of the list

ACCOUNTS_CRUD.H

Contains CRUD operations on the accounts list and file.

Imports headers DYNAMIC_ACCOUNTS.H and VALIDATION.H

void save_accounts_to_file(struct Node_accounts *head, char global_user[])

- saves all the accounts in dynamically allocated list starting at address head to file .../accounts.txt
- exception: Error opening file

struct Node_account* load_accounts(struct Node_account *head, char global_user[])

- loads all the accounts in dynamically allocated list starting at address head from file .../accounts.txt
- exception: Error opening file
- returns the current address of the start of the list

void print_all_accounts(struct Node_account *head)

- prints all the accounts from the dynamically allocated list starting at address head
- exception: No accounts available

struct Node_account* add_account(struct Node_account *head, char global_user[], char id_client[])

- adds a new account to the list starting at address head and into the .../accounts.txt file
- exception: Error opening file at path
- returns the current address of the start of the list

struct Node_account * delete_account (struct Node_account*head, char global_user[])

- deletes an account identified by id from the list starting at head and from the .../accounts.txt file
- exception: Error opening file at path, Account not found, Non-zero balance for selected account
- returns the current address of the start of the list

void check_account_balance(struct Node_account *head)

- prints the balance of the selected account from the list starting at address head

TRANSACTIONS.H

Contains the functionalities for transactions and real-life user interaction with accounts.

Imports the header ACCOUNTS_CRUD.H

void save_deposit(struct Node_account* head, char global_user[])

- saves the transactional information of a cash deposit if the account associated with the given iban is administrated by the program
- exception: Error opening the file at path

void save_withdrawal(struct Node_account *head, char user_id[], char global_user[])

- saves the transactional information of a cash withdrawal if the account associated with the given iban is administrated by the program and the user has access to it
- exception: Error opening the file at path/Permission denied

void save_transfer(struct Node_account* head, char user_id[], char global_user[])

- saves the transactional information of a transfer if the source-account associated with the given iban is administrated by the program and the user has access to it
- exception: Error opening the file at path/Permission denied

REPORTS.H

Contains the functionalities for financial reports such as account statement, transaction register and expense report.

Imports the header ACCOUNTS_CRUD.H

int check_date_in_interval(struct tm start_date, struct tm end_date, struct tm item)

- checks if argument item is between two given dates
- precondition: validity of dates
- returns 1 if the condition is met and 0 otherwise

struct tm transform_char_to_tm(char s[])

- converts the date from format "DD:MM:YYYY" to a tm object
- returns a tm datetime object
- precondition: validity of string (format and calendar-wise)

int validate_date_format(char s[])

- checks if the string provided respects the imposed "DD:MM:YYYY" format
- returns 1 if the condition is met and 0 otherwise

int validate_date(char s[])

- checks if the string provided corresponds to a valid date
- precondition: the string respects the imposed "DD:MM:YYYY" format
- returns 1 if the condition is met and 0 otherwise

int validate_second_date(struct tm end_date, struct tm start_date)

- checks if the first date comes before the second one
- precondition: both dates are valid dates
- returns 1 if the condition is met and 0 otherwise

void generate_account_statement(char global_user[], struct Node_account* head)

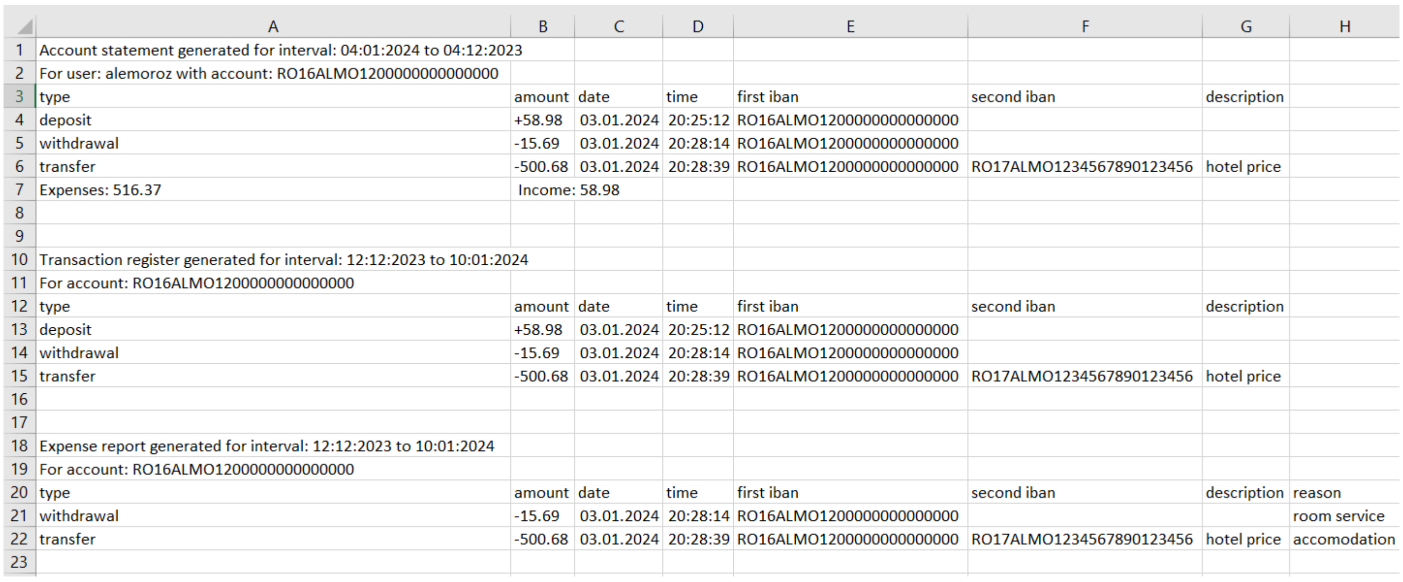
- generates an account statement containing basic information about account (user, iban), interval of time for transactions (last month) and expense & income report
- saves the report in a new file named statement-<date and time>.csv
- exception: Error opening file at path/Permission denied

void generate_transaction_register(char global_user[], struct Node_account* head)

- generates a transaction record containing basic information about account (iban), interval of time for transactions (user input)
- saves the report in a new file named transaction-<date and time>.csv
- exception: Error opening file at path/Permission denied

void generate_expense_report(char global_user[], struct Node_account* head)

- generates an expense report containing basic information about account (iban), interval of time for transactions (user input) and a short description for each expense
- saves the report in a new file named expense-<date and time>.csv
- exception: Error opening file at path/Permission denied



	A	B	C	D	E	F	G	H
1	Account statement generated for interval: 04:01:2024 to 04:12:2023							
2	For user: alemoroz with account: RO16ALMO1200000000000000							
3	type	amount	date	time	first iban	second iban	description	
4	deposit	+58.98	03.01.2024	20:25:12	RO16ALMO1200000000000000			
5	withdrawal	-15.69	03.01.2024	20:28:14	RO16ALMO1200000000000000			
6	transfer	-500.68	03.01.2024	20:28:39	RO16ALMO1200000000000000	RO17ALMO1234567890123456	hotel price	
7	Expenses: 516.37	Income: 58.98						
8								
9								
10	Transaction register generated for interval: 12:12:2023 to 10:01:2024							
11	For account: RO16ALMO1200000000000000							
12	type	amount	date	time	first iban	second iban	description	
13	deposit	+58.98	03.01.2024	20:25:12	RO16ALMO1200000000000000			
14	withdrawal	-15.69	03.01.2024	20:28:14	RO16ALMO1200000000000000			
15	transfer	-500.68	03.01.2024	20:28:39	RO16ALMO1200000000000000	RO17ALMO1234567890123456	hotel price	
16								
17								
18	Expense report generated for interval: 12:12:2023 to 10:01:2024							
19	For account: RO16ALMO1200000000000000							
20	type	amount	date	time	first iban	second iban	description	reason
21	withdrawal	-15.69	03.01.2024	20:28:14	RO16ALMO1200000000000000			room service
22	transfer	-500.68	03.01.2024	20:28:39	RO16ALMO1200000000000000	RO17ALMO1234567890123456	hotel price	accomodation
23								

DYNAMIC_USERS.H

Contains basic methods for dynamically allocating the users list.

Imports header CUSTOM_OBJECTS.H

struct Node_user* insert_at_end_user(struct Node_user* head, char id[], char username[], char password[])

- inserts a new user with given data at the end of the list starting at address head
- returns the current address of the start of the list
- exceptions: Error allocating memory for new user -> Memory not allocated! + program end

struct Node_user* delete_by_id_user(struct Node_user* head, char id[])

- deletes a user with a given id and data from the dynamically allocated list and its corresponding files
- returns the current address of the start of the list

void modify_user_by_id(struct Node_user *head, char id_string[], char pass[])

- modifies a user with a given id and data in the dynamically allocated list

ADMIN_OPERATIONS.H

Contains functionalities accesible only to the admin regarding the management of the users.

Imports the headers VALIDATION.H and DYNAMIC_USERS.H

void select_user_admin(char global_user[], char user_id[])

- selects the user for which the admin will make various operations
- exception: Error opening file at path!/Invalid username! - no user found

void add_user(char global_user[])

- creates a new user with a given username, id and password and all the required functions for it
- saves the new user in file users.csv
- exceptions: Permission denied/Error opening file at path

struct Node_user* load_users(struct Node_user *head)

- loads all the users in dynamically allocated list starting at address head from file users.csv
- exception: Error opening file at path
- returns the current address of the start of the list

void save_users_to_file(struct Node_user *head)

- saves all the users in dynamically allocated list starting at address head to file users.csv
- exception: Error opening file at path

struct Node_user * delete_user(struct Node_user *head)

- deletes a user identified by id from the list starting at head and from the users.csv file
- exception: Error opening file at path/User not found
- returns the current address of the start of the list

struct Node_user * reset_password(struct Node_user *head)

- resets the password for an user identified by id from the list starting at head and from the users.csv file
- exception: Error opening file at path/User not found

MAIN.C

Contains the console interface and menu functions.

Imports the headers TRANSACTIONS.H, CUSTOMERS_CRUD.H, REPORTS.H and ADMIN_OPERATIONS.H

void login_menu(char global_user[], char user_id[])

- login function for the app
- sets the corresponding values for global_user and user_id
- exception: invalid password -> new login/invalid username -> new login/Error opening file at path

void welcome_text()

- prints a welcome message for the user

void menu_text()

- prints a list of menu options for the user

int main()

- starting point of app
- contains the console interface which ends on user-input „exit”