**Документация по проект №1 Дневник на пътешественика**

[Линк към моя Github](https://github.com/KrashM)

**Глава 1: Увод**

***1.1*** ***Описание и идея на проекта***:

Програма, която да позволява на любителите пътешественици да споделят снимки, изживявания и впечатления. Всеки един любител може да се регистрира за да запише своите пътувания или да види преживяванията на другите за да организира свое.

***1.2*** ***Цел и задачи на разработката***:

Целта е да се напише работещо приложение с лесен за работа потребителски интерфейс. В приложението да има две основни менюта. Меню за автентикация(register/login) и основното меню за платформата. Където всеки потребител може да качва своите пътувания и съответно да разглежда пътуванията на всички останали.

Задачи:

* Да се създадат основни класове
* Да се реализират необходимите структури от данни и алгоритми
* Да се направят валидации на данните
* Да се направи структура за данните
* Да се записва цялата информация от програмата във файлове
* Да се зарежда вече запазената информация от файловете
* Да се напишат функционалности за всеки потребител, който е регистриран
* Да се направи потребителски интерфейс

***1.3 Структура на документацията***:

Документацията е структурирана по следният начин:

* Увод
* Преглед на предметната област
* Проектиране
* Реализация, тестване
* Заключение

**Глава 2: Преглед на предметната област**

***2.1.* *Основни дефиниции, концепции и алгоритми, които ще бъдат използвани***

В проектът ще бъдат използвани работа с двоични (бинарни) файлове. Обработване на данни и записването им във файлове и след това четене от там. Също ще бъдат използвани вектори за съхранение на множество от еднотипна информация. Например всички регистрирали се потребители в системата. По естество ще има файлово разделение за по-добра структура и по-добро четене на кода, както и множество от функции.

***2.2. Дефиниране на проблеми и сложност на поставената задача***

Проблемите, за които програмата е създадена да решава са:

* Създаване на потребителски акаунт
* Влизане в потребителски акаунт
* Добавяне на пътувания към потребителски акаунт
* Търсене на пътувания по дестинация

Сложността на поставената задача беше подходяща за дадения срок за нейното изпълнение.

***2.3. Подходи, методи за решаване на поставените проблемите***

За решаване на гореспоменатите проблеми, програмата използва множество функции и класове.

* Класът User полета за потребителско име, парола, имейл и множество от пътувания.
  + Функции за валидации на всички полета
  + Функция за добавяне на пътуване
  + Функция за записване в двоичен файл
  + Функция за чете от двоичен файл
* Класът Travel има полета за име на дестинацията, дата на заминаване, дата на прибиране, оценка, коментар и снимки от дестинация
  + Функции за валидации на данните
  + Функция за записване в двоичен файл
  + Функция за четене от двоичен файл
* Класът Date пази полета за ден, месец и година
* Класът Context се използва за работата на основната програма с базата данни
* Класът StringChecker е статичен клас за проверки на формата на определени стрингове като имейл или снимка например
  + Статични функции за проверка на формат
* Класът InvalidFormatException е наследник на класа std::exception който позволява за по-голяма свобода при персонализиране на грешките, които се хвърлят в програмата
* Класът Vector е стандартна имплементация на динамичен масив от произволен тип(type T)
  + Функция за автоматично преоразмеряване
  + Функции за добавяне и търкане на елемент
  + Функции за търсене на елемент

**Глава 3: Проектиране**

***3.1. Обща архитектура***

Основни класове които се използват в последствие за структурата на други класове:

* InvalidFormatException – наследява от класа std::exception
* Vector
* Date

Следващите класове са подредени по начина в който се използват един в друг:

* Travel
* User
* Context

И накрая остава основния файл на програмата, а именно Program.cpp, която контролира входа и изхода(интерфейса на програмата) и също така останалите функционалности на платформата.

***3.2. Диаграми и слоеве с най-важните извадки от кода***

* Създаване на потребител:

*User*::User(const char \*username, const char \*password, const char \*email){

    try{

*User*::validateUsername(username);

*User*::validatePassword(password);

*User*::validateEmail(email);

    }

    catch(const *InvalidFormatException* &e){ throw e; }

*this* -> username = **new** char[strlen(username) + 1];

    strcpy(this -> username, username);

*this* -> password = **new** char[strlen(password) + 1];

    strcpy(this -> password, password);

*this* -> email = **new** char[strlen(email) + 1];

    strcpy(this -> email, email);

*this* -> setPersonalDBPath();

*this* -> createUserDB();

}

* Четене на потребителите от базата данни:

void *Context*::getUsersFromDB(Vector <*User*> &users){

*ifstream* ifs("Database/users.db", *ios*::binary);

    while(ifs.peek() != EOF){

*User* temp;

        temp.read(ifs);

        users.pushBack(temp);

    }

    ifs.close();

}

* Търсене на всички пътувания до определената дестинация

Vector<*Travel*> getAllTravelToDestination(Vector<*User*> &users){

    char destination[DEFAULT\_STRING\_SIZE];

    cout << "Where do you want to go?\n: ";

    cin.getline(destination, DEFAULT\_STRING\_SIZE);

    Vector<*Travel*> result;

    for(*size\_t* i = 0; i < users.size(); i++){

        Vector<*Travel*> travels = users[i].getTravels();

        for(*size\_t* j = 0; j < travels.size(); j++)

            if(!strcmp(travels[j].getDestination(), destination))

                result.pushBack(travels[j]);

    }

    if(result.size()) return result;

    throw *InvalidFormatException*("Nobody has traveled to this destination");

}

**Глава 4: Реализация, тестване**

***Управление на паметта и алгоритми. Оптимизации.***

Програмата е сведена до минимално опростен и лесен за написване начин. Всеки клас е дефиниран със своите полета и функционалности. Като за всеки реализиран метод са изпълнени стандарти, семантики и алгоритмите им са с оптимални сложности. Всеки клас отговаря собственоръчно за управлението на паметта си. Всеки един клас който използва динамична памет в последствие я освобождава при приключването на програмата.

**Глава 5. Заключение**

***5.1. Обобщение на изпълнението на началните цели***

Програмата се справя с работа с файлове, поддържа използването на многобройни потребителски акаунти. Успешно запазване на пътувания за всеки потребител в собствена база данни. Търсене на пътуване по дестинацията.

***5.2. Насоки за бъдещо развитие и усъвършенстване***

В бъдещи версии на програмата предстои да се оптимизират структурите използвани в програмата и алгоритмите, които те използват. Така че програмата да стане като прототип, не пълна версия, на известни други програми, създадени от известни компании.