

Документация

Genickup

Съдържание

[1. Тема 2](#_Toc95894009)

[2. Автори 2](#_Toc95894010)

[3. Резюме 2](#_Toc95894011)

[3.1 Цели 2](#_Toc95894012)

[3.2 Етапи на реализация 2](#_Toc95894013)

[3.2.1 Съставяне на отбора 2](#_Toc95894014)

[3.2.2 Съставяне на график за работа 2](#_Toc95894015)

[3.2.3 Работа по проекта 2](#_Toc95894016)

[3.2.4 Презентиране 2](#_Toc95894017)

[3.3 Трудности 3](#_Toc95894018)

[3.4 Заключение 3](#_Toc95894019)

[4. Използвани функции 3](#_Toc95894020)

[4.1 BackEnd 3](#_Toc95894021)

[4.2 FrontEnd 4](#_Toc95894022)

# Тема

Темата на нашия проект е „Списък на времето“, като трябва да създадем приложение за регистриране и управление на исторически събития, реализирано чрез свързани списъци.

# Автори

* Александър Иванов Цачев – AITsachev19 – Scrum Master
* Иван Владимиров Бързев – IVBarzev19 – Back-end Developer
* Александър Ангелов Божиков – AABozhikov19 – Front-end Developer
* Стефан Огнянов Равнинов – SORavninov19 – QA Engineer

# Резюме

## Цели

Проектът ни има за цел, чрез лесен за достъп интерфейс, да даде възможността на потребителите да въвеждат и достъпват хронологично подредени исторически събития.

## Етапи на реализация

### Съставяне на отбора

За най-голяма ефективност в процеса на работа, отборът ни е съставен от участници с различни способности. Всеки се грижи за различна част от разработването на проекта:

* Scrum Master – грижи се за доброто настроение и дух в отбора, организира срещите, асистира при трудности по раздадените задачи и пише документация
* Back-end Developer – занимава се с логическата част на програмата, различните функции и обработката на данните
* Front-end Developer – работата му е свързана с интерфейса на програмата, начинът, по който се извежда менюто, резултата след изпълнение на някоя от функциите и т.н.
* QA Engineer – изработва QA документация, тества функциите в програмата, докладва при неизправности и бъгове

### Съставяне на график за работа

За да улесним процеса на работа ние съставихме график със срокове за завършване на всяка задача, която сме си поставили. За отбелязване на завършените задачи използвахме планера в GitHub.

### Работа по проекта

След по-горе споменатите стъпки дойде време за същинската работа по кода. Всеки се зае със своите задачи и периодично се срещахме в Microsoft Teams, за да обсъдим как върви процесът на работа, различни проблеми, с които сме се сблъскали, и да ги разрешим своевременно.

### Презентиране

Последния етап от разработката на проект е неговата защита.

## Трудности

По време на разработката на проекта не се срещнахме с особено много проблеми.

## Заключение

В рамките на зададения срок, създадохме ефективна и интуитивна програма

# Използвани функции

## 4.1 Back-End

|  |  |
| --- | --- |
| Име на функция | предназначение |
| NODE\* takeInput(); | иска потребителя за информация |
| void navigateToBegining(NODE\*\* node); | дава ни началото на листа |
| void navigateToEnd(NODE\*\* node); | дава ни края на листа |
| int countNodes(NODE\* node); | връща броя на елементите в листа |
| void printNode(NODE\* node); | принтира специфичен елемент от листа |
| void printAllNodes(NODE\* node); | принтира всички елементи |
| void printSpecificEraNodes(NODE\* node, std::string sEra); | принтира всички елементи от определена ера |
| void printSpecificTagNodes(NODE\* node, std::string sTag); | принтира всички елементи от определен таг |
| void sortNodesByDate(NODE\*\* node); | сортира по дата |
| void assignEra(NODE\* node); | автоматично придава ера на елемент |
| void assignAllEras(NODE\* node); | автоматично придава ера на всички елементи |
| void assignTag(NODE\* node); | автоматично придава таг на елемент |
| void assignAllTags(NODE\* node); | автоматично придава таг на всички елементи |
| void prependNode(NODE\*\* node); | поставя елемент в началото на листа |
| void appendNode(NODE\*\* node); | поставя елемент в края на листа |
| void editNode(NODE\* node); | променя определен елемент |
| NODE\* searchByTag(NODE\* start, std::string sTag); | търси следващия елемент с определен таг |
| NODE\* searchByEra(NODE\* node, std::string sEra); | търси следващия елемент с определена ера |
| void searchByTitle(NODE\* node, std::string sTitle); | търси елемент с определено заглавие |
| NODE\* searchNodeByPosition(NODE\* node, int n) | търси елемент по неговата позиция |
| void deleteNthNode(NODE\*& node, int n); | изтриване на определен елемент по позиция |
| void deleteNode(NODE\*& node); | изтриване на даден елемент |
| void deleteAllNodes(NODE\*\* node); | изтриване на всички елементи |

## 4.2 Front-End

|  |  |
| --- | --- |
| Име на функция | предназначение |
| void getBack(short int& choice); | връща се към главното меню |
| void showDeleteLogo(); | показва дизайна за страницата за изтриване |
| void showSearchLogo(); | показва дизайна за страницата за търсене |
| void showAddLogo(); | показва дизайна за страницата за добавяне |
| void showPrintLogo(); | показва дизайна за страницата за принтиране |
| void showEditLogo(); | показва дизайна за страницата за променяне |
| void showTeamLogo(); | показва дизайна за главното меню |
| void getSearchByEraDetails(NODE\* node); | иска вход име на ера и търси по тази ера |
| void getSearchByTagDetails(NODE\* node); | иска вход име на таг и търси по този таг |
| void getSearchByTitleDetails(NODE\* node); | иска вход заглави и търси даденото заглавие |
| void showAddMenu(NODE\*\* node, short int& sChoice); | показва менюто за добавяне |
| void showSearchMenu(NODE\* node, short int& sChoice); | показва менюто за търсене |
| void showDeleteMenu(NODE\* node, short int& sChoice); | показва менюто за изтриване |
| void showPrintMenu(NODE\* node, short int& sChoice); показва менюто за изтриване | показва менюто за изтриване |
| void showEditMenu(NODE\* node, short int& sChoice); | показва менюто за обработване |
| void showMainMenu(NODE\*\* node); | показва главното меню |
| void startApp(NODE\*\* node); | започва програмата |