ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ
Факултет по приложна математика и информатика

Дипломна работа

Тема: Уеб приложение за подпомагане на обучението по история.

Разработил:

Антонио Тодоров Димитров

Фак №: 471221119

Научен ръководител:

Доц. д-р инж. Малинка Иванова

ТУ – София
2023

Съдържание

- Увод.....4

Глава 1. Дефиниране на задачата……...5

Глава 2. Използвани технологии……....6

1. Spring Boot……...6
2. Java……...8
3. MySQL Database…….....9
4. HTML…….....10
5. CSS……..…...11
6. Javascript……13
7. BotPress…….....14
8. StackAI…….....….....15

Глава 3. Изисквания и софтуерна архитуктура……17

1. User-Case Диаграма……...17
2. Видове потребители……..17
   2.1. Нерегистриран потребител……....17
   2.2. Регистриран потребител……18
   2.3. Администратор….……..18
3. Функционални изисквания……...18
4. Нефункционални изисквания……...19
5. Софтуерна архитектура……19
   5.1. Компоненти……20
   5.2.Предимства…….……21

Глава 4. Реализация…….……....22

1. Модел на базата данни……22
2. Реализиране на MVC архитектурата……23
   2.1.Model……...…...23
   2.2.View……...……....24
   2.3.Controller……26
3. Изпълнение на функционалните изисквания……..27
   3.1.Регистрация……...…....28
   3.2.Вход, автентикация и проверка за администрски права……30
   3.3.Даване на администрски права и триене на профили……..32
   3.4.Интеракция с чат-бота……33

4. Изпълнение на нефункционални изисквания…….36
4.1.Сигурност…….....36
4.2.Разширяемост…….....37
4.3.Използваемост…….38

Линк към GitHub…….38
Заключение……..38

Използвана литература……..39

Увод

В съвременния свят уеб технологиите играят ключова роля в образованието и обучението на хората от всички възрасти. Създаването на интерактивни и образователни уеб приложения е станало изключително важно за подпомагането на учениците и студентите по време на техния учебен процес. Една от областите, в която уеб приложенията могат да направят значителна разлика, е обучението по история.

Историята е фундаментална част от образованието и играе важна роля в формирането на нашето разбиране за света около нас. Въпреки това, тази дисциплина често може да бъде предизвикателство за учениците поради обема на информацията, която трябва да се усвоява, и сложността на събитията и процесите през времето.

Целта на тази дипломна работа е да представи идея и разработка на уеб приложение, насочено към подпомагане на обучението по история. Това приложение ще предоставя образователни ресурси, интерактивни уроци, въпроси и тестове, които да направят процеса на учене по-забавен и ефективен.

По време на разработката на този проект ще се разгледат различни технически аспекти, включително създаването на уеб интерфейс с подходящ дизайн и лесна навигация, изграждането на база данни за съхранение на образователни материали и резултати от тестове, и интеграцията на технологии за визуализация на информация.

- **Прекалено много разпръснати материал**: В днешно време има натрупване на изключително голямо количество материали по история и то от множество различни източници. Това затруднява изключително много всеки любител на историята или просто нормалният ученик/студент, на когото му трябват всичките тези неща в компактен и лесен за четене и разбиране начин.

- **Загуба на време при търсене на информация и отговори**: За търсещите голямо количество подредена информация или среда, където да могат да получат отговори на своите въпроси, в днешно време се оказва много трудна задача. Много често имат рядко срещани въпроси, или просто голямо количество от тях и загубата на време, да четат през всичките възможни материали и прекалено нетърпима с днешните стандарти за бързодействие и научаване на информация. Приложението предоставя Изкуствен интелект, който е обучен, върху вече подредена информация и може да отговаря на всякакви въпроси относно българската история.

Глава 1. Дефиниране на задачата

За да отговаря на нуждите на потребителите, приложението трябва да бъде разработено така, че да постигне следните цели:

- **Интуитивен потребителски интерфейс:** Приложението трябва да бъде лесно за използване и навигация, да предоставя ясни инструкции и поддръжка и по този начин да улесни процеса на търсене на информация и получаване на отговор на въпросите.

- **Ефективност при търсенето:** Разработка на интелигентен изкуствен интелект, във формата на чат-бот, който да предоставя точни и релевантни по темата резултати и отговори.

- **Потребителски профил, служещ за поддържане на стари чатове:** Спестяване на време и усилия от страна на потребителят търсещ информация, като се предоставя запазване на чатовете, които са проведени с чат-бота.

- **Безопасност и защита на данните:** Трябва да се внедрят подходящи мерки за сигурност, като шифроване на данните и внимателно управление на достъпа до информацията, за да се предотврати злоупотреба или неразрешен достъп до чувствителни данни.

- **Ански интерфейс:** Трябва да се предостави и достъпен интерфейс за всички бъдещи admини, които да имат достъп до списък с всички потребители в сайта, техните имена, опцията за даване на admински статут на някой потребител и изтриването на потребители, при потенциална злоупотреба с акаунта си.

Глава 2. Използвани технологии

За изграждането на необходимия софтуер са използвани няколко различни технологии, предимно от Java и HTML. Постигането на поставените цели на задачата се случва с помощта на следните технологии:

1. Spring Boot

Spring Boot е фреймуърк за създаване на уеб приложения и микроуслуги в Java. Той е създаден, за да улесни и ускори процеса на разработка на приложения, като предоставя стандартни инструменти и настройки за разработка. Spring Boot се фокусира върху конвенциите пред конфигурацията, което означава, че по подразбиране предоставя настройки и настройки, които покриват общите случаи и позволяват на разработчиците да се концентрират върху бизнес логиката на приложението.

Някои от ключовите характеристики на Spring Boot включват:

1.1 Вграден сървър за уеб приложения: Spring Boot включва вграден сървър (например Tomcat, Jetty или Undertow), който може да бъде използван за стартиране на уеб приложенията в рамките на същия Java процес. Това намалява необходимостта от конфигурация на външен уеб сървър и улеснява стартирането и тестването на приложенията.

1.2 Автоматично конфигуриране: Spring Boot предоставя механизми за автоматично конфигуриране на приложението на базата на зависимостите и настройките по подразбиране. Този процес на автоматично конфигуриране може да бъде допълнително настроен чрез свойства и анотации.

1.3 Вградени библиотеки: Spring Boot включва много библиотеки и компоненти, които могат да бъдат използвани за създаване на различни видове приложения, включително RESTful API, уеб сървиси, синхронни и асинхронни приложения и други.

1.4 Гъвкавост и настраиваемост: Въпреки автоматичното конфигуриране, Spring Boot остава гъвкав и настраиваем. Разработчиците имат възможността да презаписват настройките по подразбиране и да настройват приложението си според нуждите си.

1.5 Висока производителност и скалируемост: Spring Boot е проектиран да предоставя висока производителност и възможност за скалиране на приложенията, което го прави подходящ за големи и натоварени проекти.

1.6 Spring Framework: Spring Boot е базиран на Spring Framework, което предоставя обширен набор от инструменти и функционалности за разработка на Java приложения. Това включва инверсия на управлението (IoC), аспектно ориентирано програмиране (AOP), работа с бази данни, сигурност и още много.

1.7 Екосистема на Spring: Spring Boot работи добре с другите проекти в екосистемата на Spring, като Spring Data за достъп до данни, Spring Security за сигурност, Spring Cloud за създаване на облачни приложения и други.

1.8 Разработка на RESTful API: С Spring Boot е лесно да създавате RESTful API, което е особено важно в съвременното програмиране. Този тип API позволява комуникация между различни компоненти и приложения.

1.9 Управление на зависимости: Spring Boot използва инструмент като Apache Maven или Gradle за управление на зависимости. Това позволява бързо и лесно добавяне на библиотеки и компоненти към вашето приложение.

1.10 Интеграция с бази данни: Spring Boot предоставя готови модули за интеграция с различни бази данни като MySQL, PostgreSQL, MongoDB и др. Това улеснява създаването на приложения, които използват данни.

1.11 Сигурност: Spring Boot има вградена поддръжка за сигурност, включително управление на потребителите, удостоверяване и авторизация.

Spring Boot е популярен сред Java разработчиците, тъй като улеснява процеса на създаване на уеб приложения и микроуслуги, като предоставя структура и инструменти, които спомагат за бързата и ефективна разработка.

2. Java

Java е мощен и широко използван програмен език, който предоставя няколко предимства в създаването на уеб приложения. Ето някои от ползите на Java в този контекст:

2.1 Портативност: Java е известен със своята портативност, което означава, че приложенията, написани на Java, могат да се изпълняват на различни операционни системи без промяна в кода. Това е особено полезно за уеб приложения, които трябва да работят на различни платформи.

2.2 Богата екосистема: Java разполага с обширна библиотека от инструменти и фреймуърки, които улесняват създаването на уеб приложения. Сред тях са Spring, JavaServer Faces (JSF), Hibernate, Apache Struts и други.

2.3 Сигурност: Java се отличава с вградена сигурност и множество механизми за защита на данните и приложенията. Това е от съществено значение за уеб приложенията, особено ако се работи с чувствителна информация.

2.4 Използване на многозадачност: Java поддържа многозадачност, което означава, че може да се обработват няколко заявки или потребители едновременно. Това е важно за уеб приложенията, които трябва да обслужват множество потребители едновременно.

2.5 Обектно-ориентиран програмен език: Java следва обектно-ориентиран подход, който улеснява структурирането и поддръжката на кода. Това прави приложенията по-модулни и лесни за разширяване.

2.6 Многоплатформеност: Java приложения могат да се изпълняват на различни устройства, включително настолни компютри, мобилни устройства и уеб сървъри. Това прави Java подходящ за широк спектър от уеб приложения.

2.7 Добра мащабируемост: Java уеб приложенията се справят добре с мащабирането, което означава, че могат да растат и да обслужват повече потребители и данни без големи промени в кода.

2.8 Интеграция с бази данни: Java има богата поддръжка за свързване и управление на бази данни, като например MySQL, PostgreSQL и Oracle. Това прави лесно съхранението и извличането на данни за уеб приложения.

2.9 Скорост и ефективност: JVM (Java Virtual Machine) е оптимизиран за изпълнение на бърз и ефективен код. Това означава, че Java уеб приложенията могат да бъдат бързи и да осигуряват добра производителност.

2.10 Модулност и реусабилност: Java насърчава модулността на кода, която улеснява реусабилността на компонентите. Този подход прави разработката по-ефективна и бърза.**Обновления и сигурност:** Java често обновява своята сигурност и предоставя поправки за известни уязвимости. Това гарантира по-висока сигурност на уеб приложенията.

2.11 Богати възможности за визуализация: Java предоставя библиотеки и фреймуърки за създаване на интерактивни и визуално привлекателни уеб интерфейси.

2.12 Поддръжка на множество протоколи: Java поддържа различни мрежови протоколи като HTTP, HTTPS, WebSocket и други, което позволява създаването на разнообразни уеб приложения.

3. MySQL Database

MySQL е мощна релационна система за управление на бази данни (RDBMS), която се използва широко в уеб разработката поради своите множество предимства:

3.1 Скорост и ефективност: MySQL е известен с бързината и ефективността на заявките си. Това прави базите данни, използващи MySQL, подходящи за уеб приложения, където отговорът на заявките трябва да бъде бърз.

3.2 Скелетна структура: MySQL предоставя структура на таблиците и релации между тях, което улеснява организацията и съхранението на данни в уеб приложенията.

3.3 Сигурност: MySQL има много високи стандарти за сигурност, което го прави подходящ избор за уеб приложения, особено ако се използват чувствителни данни.

3.4 Множество поддържани езици за програмиране: MySQL може да бъде интегриран с много различни програмни езици, включително PHP, Java, Python и

други. Това позволява на разработчиците да избират най-подходящият език за техните нужди.

3.5 Операционна система и архитектура: MySQL е съвместим с много различни операционни системи и архитектури, което го прави много гъвкав и универсален за разработка на уеб приложения.

3.6 Скалируемост: MySQL поддържа възможността за скалиране на базите данни, което го прави подходящ за уеб приложения с висок трафик.

3.7 Безплатно и с отворен код: MySQL се предоставя като безплатен и с отворен код софтуер, което намалява разходите и позволява на разработчиците да го променят според нуждите си.

3.8 Общност и поддръжка: MySQL има голяма общност от потребители и разработчици, което означава, че има множество ресурси и поддръжка за него онлайн.

3.9 Подходящ за малки и големи проекти: MySQL може да бъде използван както за малки уеб сайтове, така и за големи и сложни уеб приложения.

3.10 Инструменти за управление: MySQL предлага различни инструменти за управление и администрация на базите данни, което улеснява работата на администраторите и разработчиците.

4. HTML (Hyper Text Markup Language)

HTML е стандартен език за маркиране, който се използва за създаване и структуриране на уеб страници. HTML позволява на разработчиците да дефинират структурата и съдържанието на уеб страниците чрез използването на специални елементи и тагове. В едно уеб приложение HTML има няколко важни ползи:

4.1 Структуриране на Съдържание: HTML позволява ясно да се дефинира структурата на уеб страницата. Това включва заглавия, параграфи, списъци, таблици и други елементи, които улесняват представянето на информацията.

4.2 Семантика: HTML предлага семантични елементи, които описват типа на съдържанието. Например, тагът <header> се използва за обозначаване на заглавния блок на уеб страницата, а <article> указва наличието на статия или новина.

4.3 Визуализация: HTML работи с CSS (Cascading Style Sheets), което позволява стилизиране и оформление на уеб страниците. Това включва цветове, шрифтове, размери и други стилове.

4.4 Интерактивност: HTML може да съдържа форми, бутони и хипервръзки, които позволяват на потребителите да взаимодействат с уеб приложението.

4.5 Съвместимост: HTML е стандарт, който е поддържан от всички модерни браузъри. Това означава, че уеб приложението ви ще бъде достъпно за голям брой потребители.

4.6 Достъпност: Правилното използване на HTML тагове и атрибути може да подобри достъпността на уеб страницата за хора с увреждания, като например слепи потребители.

4.7 SEO (Search Engine Optimization): HTML с правилната структура може да подобри видимостта на уеб страницата в търсачките.

4.8 Интеграция с други Технологии: HTML може да бъде комбиниран с JavaScript за добавяне на интерактивност и със CSS за стилизиране. Този комбиниран подход позволява разработката на по-сложни и функционални уеб приложения.

Използвайки HTML като основен строителен блок на уеб приложенията, разработчиците могат да създадат атрактивни, функционални и лесно достъпни уеб сайтове и приложения.

5. CSS (Cascading Style Sheets)

CSS е стилов език, който се използва за стилизиране и форматиране на уеб страници и уеб приложения. CSS играе ключова роля в подобряването на външния вид и изживяването на потребителите при работа с уеб приложения. Ето някои от ползите на CSS в уеб разработката:

5.1 Визуален Дизайн: CSS позволява на разработчиците да задават цветове, шрифтове, размери на текст, отстъпи, рамки, фонове и други визуални ефекти. Това прави възможно създаването на атрактивни и професионално изглеждащи уеб страници.

5.2 Разделение на Съдържание и Дизайн: С CSS, стиловете се отделят от HTML съдържанието. Този подход, известен като разделение на съдържание и представление, прави управлението на дизайна по-лесно и поддържаемо.

5.3 Подвижност: Промяната на стиловете в CSS е лесна и бърза. Това позволява на разработчиците да променят визуалния дизайн на цялото уеб приложение с минимален труд.

5.4 Адаптивен и Отзивчив Дизайн: CSS позволява създаването на адаптивни и отзивчиви дизайни, които се приспособяват към различни размери на екрана и устройства (например, мобилни телефони, таблети, десктоп компютри).

5.5 Браузърна Съвместимост: CSS е поддържан от всички модерни браузъри, което осигурява съвместимост и консистентност на визуалния дизайн на уеб страниците.

5.6 SEO Оптимизация: Правилната употреба на CSS може да подобри SEO (Search Engine Optimization) резултатите на уеб страниците чрез оптимизация на структурата и структурата на уеб страниците.

5.7 Ефективност: CSS позволява на разработчиците да стилизират множество страници чрез преизползване на стилове. Това намалява дублирането на код и прави уеб приложението по-ефективно.

5.8 Анимации и Транзиции: CSS предоставя възможности за създаване на анимации и транзиции, които могат да подобрят визуалния опит на потребителите.

5.9 Достъпност: С правилната употреба на CSS, разработчиците могат да подобрят достъпността на уеб приложението за хора с увреждания.

5.10 Интеграция: CSS може да бъде лесно интегриран с други технологии като HTML, JavaScript и дори CSS препроцесори като Sass и Less.

Общо казано, CSS е мощен инструмент, който позволява на уеб разработчиците да създават уеб приложения с впечатляващ и интуитивен дизайн, като същевременно улеснява поддръжката и оптимизацията им.

6. Javascript

JavaScript (JS) е широко използван програмен език за уеб разработка, който предоставя много ползи в създаването на уеб базирани приложения. Ето някои от тези ползи:

6.1 Интерактивност: JavaScript позволява добавянето на интерактивност към уеб приложението. Този език позволява на потребителите да взаимодействат с уеб страниците, да извършват действия като кликване върху бутони, попълване на форми и други.

6.2 Асинхронност: JavaScript поддържа асинхронни заявки към сървъра, което позволява на уеб приложението да работи бързо и ефективно. Това е важно за зареждането на данни без да блокира интерфейса на потребителя.

6.3 Манипулация на DOM: JavaScript предоставя възможност за манипулиране на DOM (Document Object Model) на уеб страницата. Това позволява динамичната промяна на съдържанието и структурата на уеб страницата.

6.4 Валидация на данни: JS може да се използва за валидация на данни, въведени от потребителите във форми преди тяхното изпращане към сървъра. Това помага за предотвратяване на грешки и подобрява сигурността на приложението.

6.5 Анимации и графика: JavaScript позволява създаването на анимации и визуални ефекти на уеб страниците, което може да подобри потребителския интерфейс и изживяването.

6.6 Интеграция с различни технологии: JS се интегрира лесно с HTML и CSS, което позволява създаването на цялостни уеб приложения. Освен това, той може да работи с различни библиотеки и фреймуърки като React, Angular и Vue.js.

6.7 Крос-платформеност: JavaScript може да се използва както за уеб приложения, така и за мобилни приложения чрез фреймуърки като React Native и Ionic.

6.8 Обработка на събития: JS позволява лесната обработка на събития като кликване, навигация, въвеждане на клавиши и други. Това прави възможно реагирането на приложението на потребителските действия.

6.9 Разширяемост: Множество библиотеки и модули на разположение направо от общността направляват разработката на уеб приложения по-лесна и бърза.

6.10 Отворен код и общност: JavaScript е отворен код и разполага с голяма общност от разработчици, което означава, че можете да намерите множество ресурси, библиотеки и форуми за поддръжка.

Тези ползи правят JavaScript ключов инструмент в уеб разработката и го правят подходящ за създаването на динамични и интерактивни уеб приложения.

7. Botpress

Botpress е отворена платформа за разработване и управление на чат ботове. Това приложение е често използвано за създаване на чат ботове, които могат да отговарят на въпроси на потребителите, да изпълняват определени задачи и да се интегрират с различни системи и приложения.

Ето някои от основните характеристики и възможности на Botpress за уеб приложения:

7.1. Гъвкавост при разработване: Botpress предоставя множество инструменти и библиотеки за създаване на персонализирани чат ботове. Можете да настроите бота да отговаря на конкретни въпроси и да изпълнява определени задачи.

7.2. Множество канали за комуникация: Botpress позволява интеграция с различни канали за комуникация, като уеб чатове, Facebook Messenger, Slack и други. Това ви позволява да се свързвате с потребителите в техния предпочитан канал.

7.3. Изкуствен интелект и машинно обучение: Botpress разполага с интеграция с изкуствен интелект и машинно обучение, което позволява на бота да учи и да се подобрява с времето, като разпознава и отговаря на по-добре на въпросите на потребителите.

7.4. Интеграция с външни системи: Можете да интегрирате бота с различни външни системи и приложения, като например бази данни, CRM системи или API, за да извличате и обработвате информация.

7.5. Анализ и статистика: Botpress предоставя инструменти за анализ и статистика, които ви помагат да следите как се използва бота и да подобрите неговата ефективност.

7.6. Гъвкавост при интеграцията: Botpress е гъвкава и мащабируема платформа, която позволява на разработчиците да персонализират и разширяват функционалността й.

Botpress е подходящ за разработчици, които искат да създадат и управляват чат ботове в уеб приложения с разнообразни функционалности и интеграции. Тази платформа предоставя мощни инструменти за създаване на различни видове ботове, отговарящи на конкретни нужди и изисквания на потребителите.

8. StackAI

Най-бързият и най-надежден начин за интегриране на персонализирани Големи Езикови Модели (ГЕМ), като например ChatGPT, във всеки един продукт или екип.

Използвайки мощността на изкуственият интелект за всякакви приложения с Stack AI, той инструмент без необходимост от програмиране, който позволява да се проектират, тестват и внедряват AI работни потоци, чрез използване на модели като ChatGPT. Тази платформа позволява на екипи да създават разнообразни приложения, включително:

- **Чатботове и асистенти:** За взаимодействие с потребители, отговаряне на въпроси и изпълняване на задачи, използвайки ваши готови вътрешни данни и API-ове.

- **Обработка на документи:** За извличане на изводи, предоставяне на резюмета и отговаряне на въпроси от всеки документ, независимо от дължината му.

- **Отговаряне на въпроси върху бази данни:** Можете да свързвате модели като ChatGPT с бази данни като Notion, Airtable или Postgres, за да получите ценни изводи за вашите проекти.

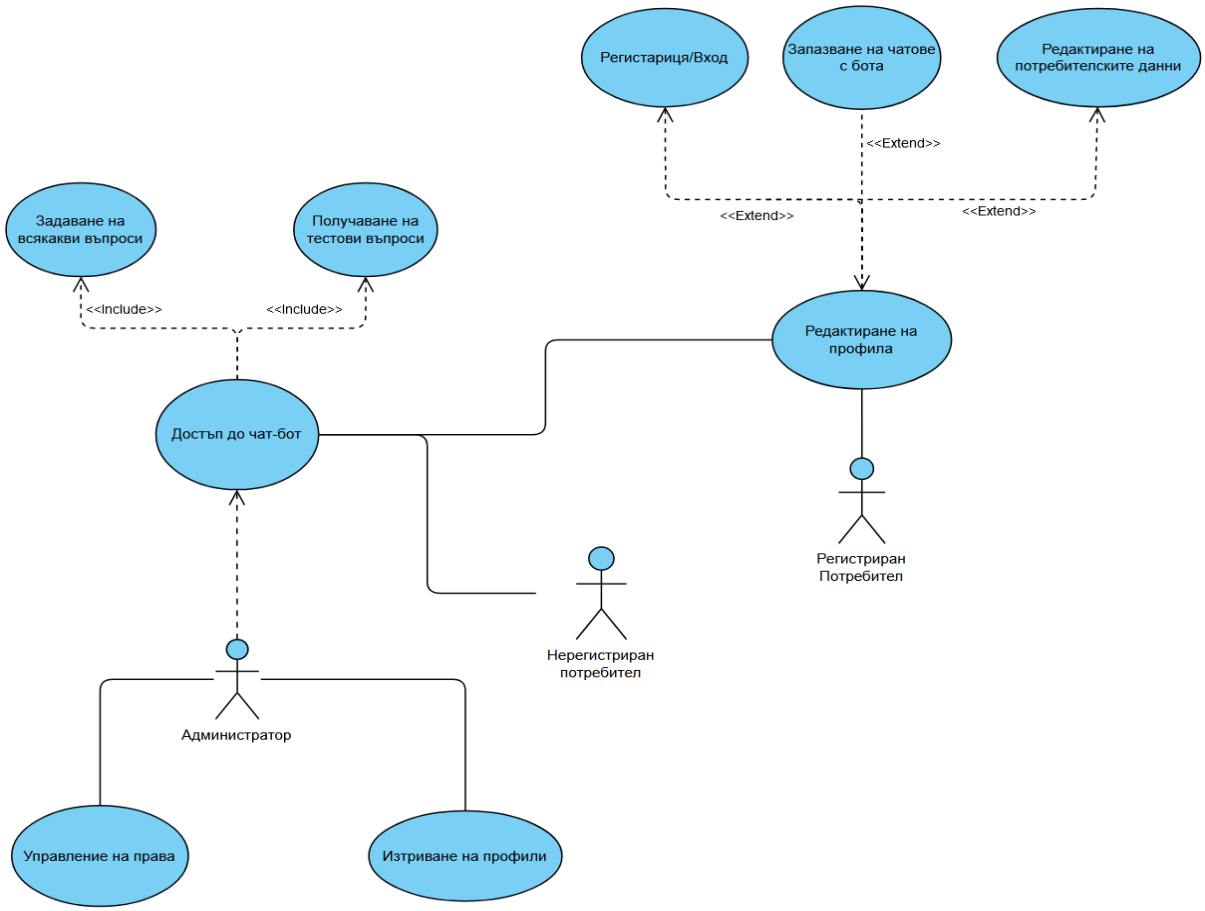
- **Създаване на content:** Има полза да генерира тагове, резюмета и лесно да се прехвърлят стилове или формати между документи и източници на данни.

Stack AI ви помага да интегрирате Големи Езикови Модели (ГЕМ) с вашето приложение за минути.

Интерфейсът без код ви позволява визуално да свържете входове, изходи, ГЕМ-ове, векторни бази данни и зареждачи на документи, за да създадете бързо изкуствен интелектен за поток от задачи като: разговорен ИИ, отговаряне на въпроси, обработка на документи, създаване на съдържание и други...

Глава 3. Изисквания и софтуерна архитектура

1. Use-case диаграма



Фиг. 1 – Use-Case диаграма на приложението

В приложената диаграма са онагледени основните сценарии при работата с приложението от гледната точка на трите възможни роли за потребителите – нерегистриран user, регистриран user и администратор.

2. Видове потребители

2.1. Нерегистриран потребител

Тази роля е по-подразбиране, като такъв потребител бива, всеки, който все още не си е създал профил в уеб приложението. Той ще има достъп до разговор с чат-бота и възможността да се регистрира в сайта по всяко едно

време, но няма да има достъп до другите услуги, които може да предостави самото приложение.

2.2. Регистриран потребител

С тази роля, всеки един човек, който вземе решението да направи своята регистрация в уеб приложението, си прави голяма услуга от към спестяване на ресурси за всички бъдещи диалози, които ще проведе. Единствените неща, за които той като потребител ще трябва да внимава, са името на неговият или нейният акаунт за това какви разговори провежда с чат-бота, тъй като някой от администраторите, може да прецени, че този акаунт не влиза в рамките.

2.3. Администратор

Ролята „администратор“ е предвидена за сравнително малък брой доверени лица, отговорни за администрирането на профилите в приложението. Единствено администраторите имат правата да правят други нови администратори, при предварително посъветване с мнозинството от останалите администратори, за да не се получава огромна инфлация в бройката на хора с права в това приложение. Администраторите, също могат да се разполагат с правото да изтрият нечий акаунт при нарушение на някои от правилата за именуване на акаунт или при опасения за некоректно използване на чат-ботът.

3. Функционални изисквания

Както става ясно от use-case диаграмата, общите стъпки, през които трите вида потребители преминават са, достъп до самият сайт и използването на чат-ботът. От тук вече се разделят, като понеже администраторът и регистрираният потребител имат вече съществуващи акаунти, то те следва да имат достъп до своят профил и правенето на редакции по него, като нека не забравяме и опцията да излязат от своя профил. И накрая остава администраторът, който би следвало да има отделна страница, където да получи списък с всички регистрирани потребители, с чиито акаунти да може да се разполага и да използва своите правомощия при необходимост.

От описаното до сега, следва да се наложат следните конкретни функционални изисквания:

Общи:

- Вход
- Автентикация
- Регистрация
- Промяна на потребителско име
- Комуникация с чат-бота

Администраторски:

- Изтриване на профили
- Задаване на админски права на други потребители

4. Нефункционални изисквания

За да се изпълняват очакванията за качествена работа, софтуерната система трябва да отговаря на следните нефункционални изисквания:

- **Сигурност -** Трябва да има висока степен на защита на данните и ключовите функционалности от неправомерен достъп. Трябва да се осигури идентификация, автентикация и оторизация на потребителите. Паролите трябва да се съхраняват в хеширан вид.

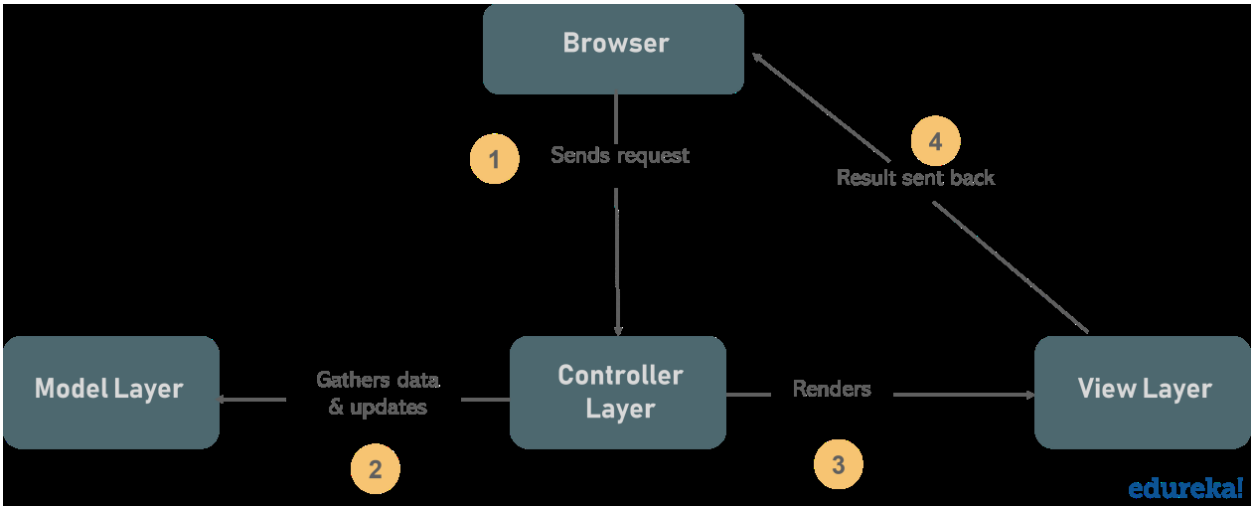
- **Разширяемост –** Осигуряване на бъдещо развитие развитие и растеж. Цели се както добавянето на нови функционалности, така и модифицирането на стари да изисква възможно най-малко усилия и промени по вече съществуващи софтуерни компоненти.

- **Използваемост -** Поради насочеността на приложението да обслужва хора от всякакви сфери, се очаква средностатистическият потребител да бъде поне леко технически грамотен. Поради тази причина използваемостта се нарежда зад останалите нефункционални изисквания, но все пак се зачита като важен атрибут на качеството. Лекотата при работа и удобният интерфейс биха могли да направят приложението предпочитано пред конкуретните софтуерни продукти.

5. Софтуерна архитектура

Разработеното приложение следва принципите на MVC (Model-View-Controller) архитектурата- широко разпространен архитектурен шаблон в областта на

уеб базираните технологии. Характеризира се с разделянето на приложението на три основни компонента - модел (Model), изглед (View) и контролер (Controller), които работят заедно, за да постигнат разделение на отговорностите и да осигурят лесна поддръжка, разширяемост и преизползваемост на кода.



Фиг. 2 – Схема на MVC архитектура (източник: edureka.co)

5.1. Компоненти

Трите компонента, съставящи тази архитектура, имат следните характеристики:

- **Модел:** Представлява данните и бизнес логиката на приложението. Той съхранява информацията, обработва заявки и предоставя методи за достъп и манипулиране на данните. Моделът не зависи от изгледа или контролера и служи като едностранен източник на данни за приложението.
- **Изглед:** Отговорен е за визуализацията на данните и представянето им на потребителя. Той предоставя графичен интерфейс, чрез който потребителят може да взаимодейства с приложението. Изгледът не съдържа бизнес логика и се основава на данните, предоставени от модела. Той може да бъде променен без да се засяга логиката на модела или контролера.
- **Контролер:** представлява връзката между модела и изгледа. Той приема заявки от потребителя, обработва ги и управлява взаимодействието между модела и изгледа. Контролерът съдържа логиката на приложението и определя какви действия да се предприемат в отговор на заявките. След обработката на заявката, контролерът обновява модела и избира подходящ изглед за показване на промените.

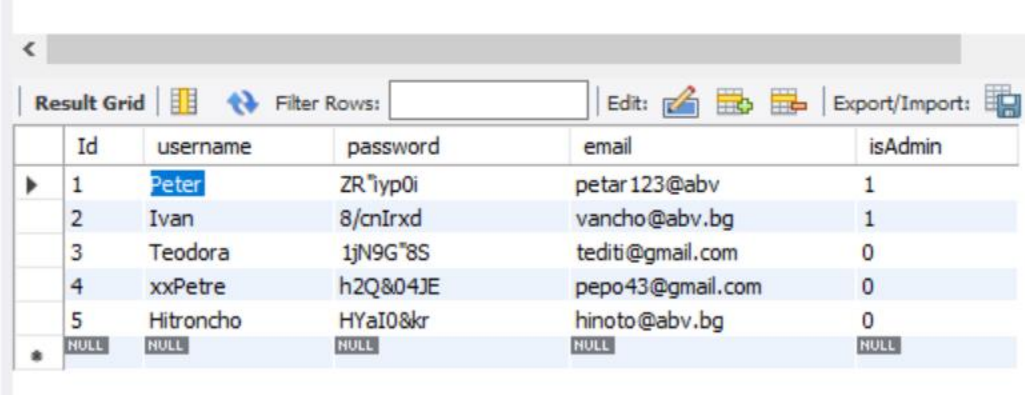
5.2. Предимства

Използването на MVC архитектура дава следните предимства:

- **Разделение на отговорностите:** Всеки компонент има ясно определени роля и отговорности. Моделът се грижи за данните и бизнес логиката, изгледът се грижи за визуализацията, а контролерът управлява потока на данни и действията на потребителя. Това позволява по-добра организация на кода и лесно поддържане на приложението.
- **Висока степен на разширяемост:** MVC архитектурата предоставя гъвкавост при добавянето на нови функционалности или промяна на съществуващите. Тъй като компонентите са добре разделени, може да се добавят нови модели, изгледи или контролери, без да се засягат останалите части от приложението.
- **Повторно използване на компонентите:** Понеже всички компоненти извършват своя специфична функция, те могат да бъдат използвани повторно в различни части на приложението. Пример за това предимство би могло да бъде използването на един и същ модел в различни изгледи или пък прилагането на обща логика за обработка на заявки на контролерите.

Глава 4. Реализация

1. Модел на базата данни



Id	username	password	email	isAdmin
1	Peter	ZR"iyp0i	petar123@abv	1
2	Ivan	8/cnIrxd	vancho@abv.bg	1
3	Teodora	1jN9G"8S	tediti@gmail.com	0
4	xxPetre	h2Q&04JE	pepo43@gmail.com	0
5	Hitroncho	HYaI0&kr	hinoto@abv.bg	0
NULL	NULL	NULL	NULL	NULL

Фиг. 3 – Модел на базата данни за приложението

На приложената схема е изобразена следната таблица:

- **Users –** регистрирани потребители в приложението
  - **Id –** Първичен ключ в текстови формат, за да се записва под формата на GUID(Глобално уникален идентификатор) и да позволява по-голям брой записи от INT.
  - **Username –** Потребителското име на регистрираният потребител, което трябва да спазва Namin convention, за коректност на името.
  - **Password -** Паролата
  - **Email –** електронна поща на потребителя, с която се регистрира и на която в бъдеще ще може да получава известия.
  - **IsAdmin –** Променливата, която казва, дали някой от записаните в системата потребители имат административни права и съответно достъп до административното меню.

2. Реализиране на MVC архитектурата

Трите компонента Model, View и Controller са разделени на отделни части в рамките на проекта:

2.1. Model

Както може да се види от снимките отдолу, моделът е изграден на **Java** и може да се види структурата на моделът на потребителите. Освен това от кода, може да се забележи, как се изгражда връзката между **Java** кодът и базата данни, която се използва. Пример за това биват: @Table, което задава името на таблицата в базата данни; @Column, което задава името на дадената колона в таблицата и @Id, заедно с @GeneratedValue, които задават ID-то, като ключова стойност в таблицата и задават автоматичното му генериране.

```
1   package com.example.javaprojectmain;
2
3   import javax.persistence.Column;
4   import javax.persistence.Entity;
5   import javax.persistence.GeneratedValue;
6   import javax.persistence.GenerationType;
7   import javax.persistence.Id;
8   import javax.persistence.Table;
9
    18 usages
10  @Entity
11  @Table(name = "siteusers")
12  public class User {
13      @Id
14      @Column (
15              name = "Id",
16              nullable = false
17      )
18      @GeneratedValue(
19              strategy = GenerationType.IDENTITY
20      )
21      private int id;
        3 usages
22      @Column(
23              name = "username",
24              nullable = false
25      )
26      private String username;
        3 usages
27      @Column(
28              name = "password",
29              nullable = false
30      )
31      private String password;
        3 usages
32      @Column(
33              name = "email",
34              nullable = false
35      )
```

Фиг. 4.1. – class User – структура на модела

```java
    private String email;
    2 usages
    @Column(
            name = "isAdmin"
    )
    private boolean IsAdmin;

    public User() {
    }

    no usages
    public User(String username, String password, String email, boolean isAdmin) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.IsAdmin = isAdmin;
    }

    3 usages
    public String getUsername() {
        return this.username;
    }

    no usages
    public void setUsername(String username) { this.username = username; }

    no usages
    public int getId() { return this.id; }

    no usages
    public void setId(int id) { this.id = id; }

    2 usages
    public String getPassword() { return this.password; }

    no usages
    public void setPassword(String password) { this.password = password; }

    no usages
    public String getEmail() { return this.email; }
```

Фиг. 4.2. - class User – структура на модела

2.2. View

В следващите снимки могат да се видят всички файлове, които изграждат графичният интерфейс. След преглеждане на снимките, може да се забележи, че той е изграден от html, css и javascript файлове, които са силно взаимосвързани, поради което, е по-добре да бъдат в отделна папка.

Name	Date modified	Type
AboutUs	9/10/2023 7:48 PM	Firefox HTML Docum...
AdminMenu	9/10/2023 7:48 PM	Firefox HTML Docum...
background	1/17/2023 9:54 PM	JPG File
ContactInfo	9/10/2023 7:48 PM	Firefox HTML Docum...
HomePage	9/10/2023 7:48 PM	Firefox HTML Docum...
LoginPage	9/10/2023 7:29 PM	Firefox HTML Docum...
ProfilePage	9/10/2023 9:14 PM	Firefox HTML Docum...
script	9/11/2023 2:02 PM	JavaScript Source File
styleSheet	9/10/2023 8:41 PM	CSS File

Фиг. 5 – Структура на View компонента

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <title>История за всички</title>
    <link rel="stylesheet" href="styleSheet.css">
</head>

<body>
    <header>
        <h2 class="logo">История за всички!</h2>
        <nav class="navigation">
            <a href="HomePage.html" class="Nav-Btn">Начало</a>
            <a href="#" class="Nav-Btn">За сайта</a>
            <a href="ContactInfo.html" class="Nav-Btn">Контакти</a>
            <a href="AdminMenu.html" class="Nav-Btn" id="AdminBtn">Admin</a>
            <a href="LoginPage.html">
                <button class="btnLogin-popup">Влизане</button>
            </a>
        </nav>
    </header>

    <div class="main-wrapper">
        <div class="main-box">
            <h2>Привет Приятели!</h2>
            <p>Това е уеб базирана система, изградена около централната идея за
                чат бот с изкуствен интелект, който да помага на ученици и студенти
                по всякакъв възможен начин. Проектът принадлежи на Антонио Тодоров
                Димитров и е изграден с цел дипломна работа за "Технически университет" София.
            </p>
        </div>
    </div>

    <script type="module" src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
    <script nomodule src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"></script>
    <script src="script.js"></script>
    <script src="https://cdn.botpress.cloud/webchat/v0/inject.js"></script>
    <script>
     window.botpressWebChat.init({
      "composerPlaceholder": "Chat with Петър",
      "botConversationDescription": "Историк оптимист тук да помогне!",
```

Фиг. 6 – Примерен View html файл

Като пример на фиг.6 е даден изгледът на прозореца “AboutUs”, в който всеки един потребител, може да прочете повече за идеята и целта на това уеб базирано приложение и причината, поради, която е създадено. Освен това, както ще се види и на всяка една друга страница в уеб приложението, има script, който зарежда чат-ботът, за да може независимо от това, на коя страница, се намира даденият потребител, да има достъп до това да проведе чат с него и да го пита въпроси или да получава тестови въпроси.

2.3. Controller

```java
package com.example.javaprojectmain;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.mindrot.jbcrypt.BCrypt;

import java.util.List;

@RestController
@RequestMapping("/siteusers")
public class UserController {

    @Autowired
    private UserService userService;
    1 usage
    String StoredSalt = "Random Passwords"; //Temporarily preset

    @PostMapping("/addUser")
    public boolean createUser(@RequestBody User user) {
        User createdUser = userService.createUser(user);
        String hashedPassword = BCrypt.hashpw(user.getPassword(),StoredSalt);

        return createdUser != null;
    }

    @PostMapping("/authUser")
    public boolean authenticateUser(@RequestBody User user) {

        return userService.authenticateUser(user.getUsername(), user.getPassword());
    }

    @PostMapping("/adminCheck")
    public boolean adminCheck(@RequestBody User user) {

        return userService.adminCheck(user.getUsername());
    }
```

Фиг. 7 – Основният User Controller

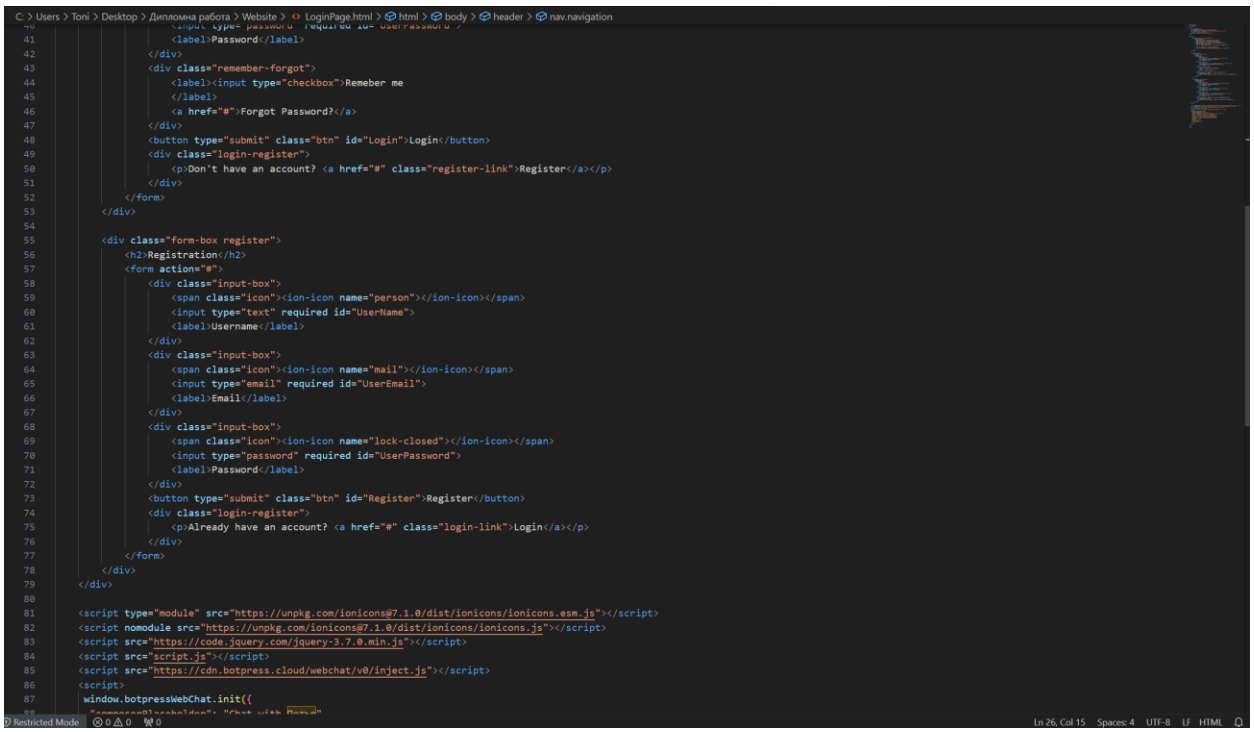
Тук можем да видим основният controller, където се съчетават всички методи за достъп и съхранение на информацията от базата данни на MySQL. Създадени са класовете UserRepository, който е връзката между базата данни и кода, и UserService, който прави транзицията между репозитори класа и контролер класа. А в случаят, контролер класът е този, който поема HTTP Request-ове и изпълнява конкретните фукнкции в зависимост от Request-а.

3. Изпълнение на функционалните изисквания

Тук ще бъде описана реализацията на ключовите изисквания:

3.1. Регистрация

Регистрацията се случва, когато от началното меню или някое от другите менюта, се натисне бутон “Login”, където получаваме форма за регистрация. И визуалната част се изпълнява чрез този html код:



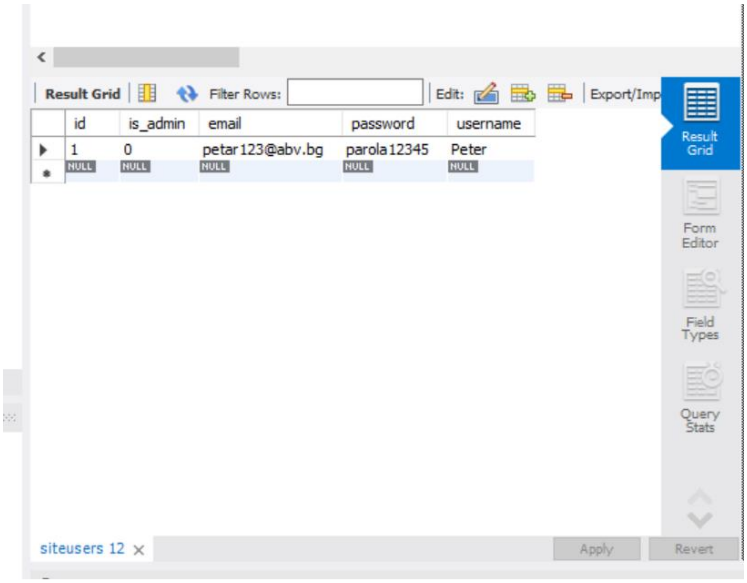
Фиг. 8 – Html код на регистрационна форма

След това имаме javascript кодът, който изпраща HTTP заявка към Java кода, със информацията, която е получил за потребителят, който се опитва да се регистрира и съответно ще излезе Pop-up прозорец, който ще извести дали регистрацията е била успешна или не. Създаването на самата заявка се получава чрез използването на $.ajax({}) метода, който взима конкретният URL и директно я създава и изпраща към сървър частта.

```javascript
function createUser() {
    const userName = document.getElementById('UserName').value;
    const userPassword = document.getElementById('UserPassword').value;
    const userEmail = document.getElementById('UserEmail').value;
    const userIsAdmin = 0;

      $.ajax({
        url: "http://localhost:8080/siteusers/addUser",
        type: "POST",
        crossDomain: true,
        data: JSON.stringify({
            username: userName,
            password: userPassword,
            email: userEmail,
            isAdmin: userIsAdmin
        }),
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function(data){
            if(data == false)
            {
                popupContainer.classList.add('.active');
                var RegisterHeader = document.getElementById("Popup-Header");
                var RegisterPara = document.getElementById("Popup-Paragraph");

                RegisterHeader.textContent = "Неуспешна регистрация!";
                RegisterPara.textContent = "Моля затворете този прозорец и опитайте отново!";

                closeBtn.onclick = () => {
                    popupContainer.classList.remove('.active');
                    RegisterHeader.textContent = "";
                    RegisterPara.textContent = "";
                }
            }
            else {
                popupContainer.classList.add('.active');
                var RegisterHeader = document.getElementById("Popup-Header");
                var RegisterPara = document.getElementById("Popup-Paragraph");

                RegisterHeader.textContent = "Успешна регистрация!";
                RegisterPara.textContent = "Моля затворете този прозорец и влезте в профила си!";

                closeBtn.onclick = () => {
                    popupContainer.classList.remove('.active');
                    RegisterHeader.textContent = "";
                    RegisterPara.textContent = "";
                }
```

Фиг. 9 – Javascript код за създаване на User

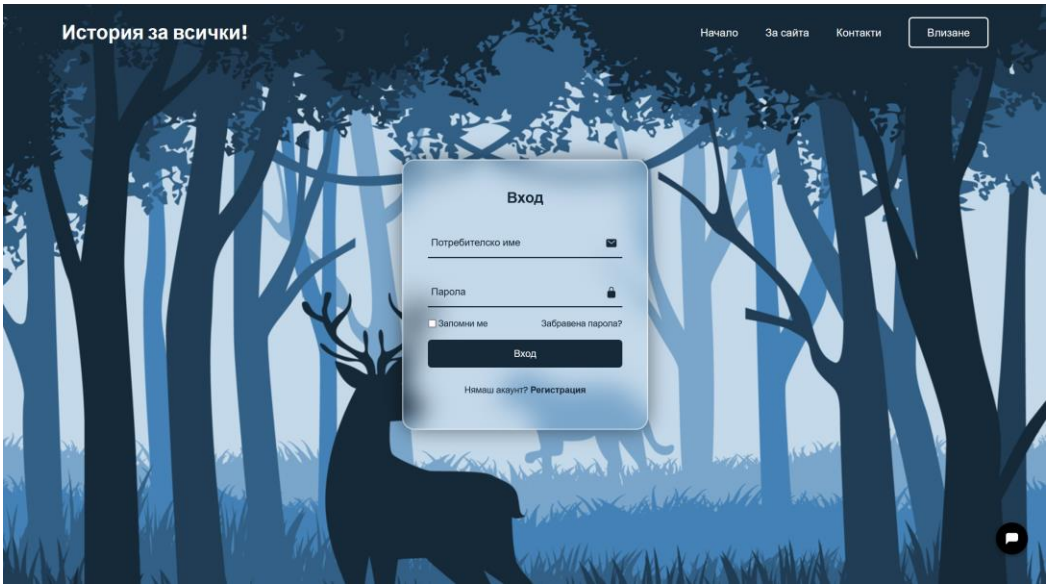
Съответно след това заявката от Javascript класа, се получава от Java кода, който проверява какъв тип заявка е пусната и заедно с това, коя функционалност се иска да бъде изпълнена. Като тук се извиква createUser функцията, която взима цялата получена информация от заявката, създава връзката с базата данни на MySQL. От там вече се изпълнява проверката и сравнението на получената информация с базата данни, по username-а на потребителите. Вече имайки резултатите от това, се връщаме и казваме на цялата връзка от кодове, какво се е случило и те преценяват какво да направят. От фиг. 7 може да се види и как по-точно се случва регистрацията/създаването на новият потребител в Java.



Фиг. 10 – Примерна база данни с един запис в MySQL

3.2 Вход, автентикация и проверка за admински права

Входът и автентикацията на потребителите се случва наведнъж. Първоначално се приема информацията от front-end-а, където има подготвени input-box-ове. Така от html кода, тя се препраща към javascript кода, който може да се види на фигурата след това.



Фиг. 11 – Логин страница в уеб приложението

```
<div class="wrapper">
    <div class="form-box login">
        <h2>Login</h2>
        <form action="#">
            <div class="input-box">
                <span class="icon"><ion-icon name="mail"></ion-icon></span>
                <input type="text" required id="UserName">
                <label>Username</label>
            </div>
            <div class="input-box">
                <span class="icon"><ion-icon name="lock-closed"></ion-icon></span>
                <input type="password" required id="UserPassword">
                <label>Password</label>
            </div>
            <div class="remember-forgot">
                <label><input type="checkbox">Remeber me
                </label>
                <a href="#">Forgot Password?</a>
            </div>
            <button type="submit" class="btn" id="Login">Login</button>
            <div class="login-register">
                <p>Don't have an account? <a href="#" class="register-link">Register</a></p>
            </div>
        </form>
    </div>
```

Фиг. 12 – Html код за Login частта от уеб приложението

Тук автентикацията се случва чрез $.ajax({}) функционалността, което, както беше описано и при регистрацията, се създава HTTP заявката, която изпраща действието да Java кода и от там получаваме response, дали потребителят е автентикиран успешно и след това по подобен начин прави проверката и дали потребителят е админ, с което задава неговите права да вижда и има достъп до админското меню, където може да изтрива потребители или да ги зададе като други администратори.

```javascript
function authenticateUser() {
    const userName = document.getElementById('UserName').value;
    const userPassword = document.getElementById('UserPassword').value;

    const user = {
        userName,
        userPassword
      };

      $.ajax({
        url: "http://localhost:8080/siteusers/addUser",
        type: "POST",
        crossDomain: true,
        data: JSON.stringify({
            username: userName,
            password: userPassword
        }),
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function(data){
            if(data == true)
            {
                CurrentUser.UserName=userName;
                btnLogin.style.display = "none";
                adminCheck(user);
                btnProfile.style.display = "block";
                //Change Login Button to Profile Button
            }
            else {
                popupContainer.classList.add('.active');
                var RegisterHeader = document.getElementById("Popup-Header");
                var RegisterPara = document.getElementById("Popup-Paragraph");

                RegisterHeader.textContent = "Неуспешно влизане в профила!";
                RegisterPara.textContent = "Моля затворете този прозорец и опитайте отново!";

                closeBtn.onclick = () => {
                    popupContainer.classList.remove('.active');
                    RegisterHeader.textContent = "";
                    RegisterPara.textContent = "";
                }
            }

        },
        error: function(error){
            console.error("Request failed:" + error.statusText);
        }
      })
```

Фиг. 13 – Javascript код за автентикацията/вход, заедно с извикване с проверка за админ

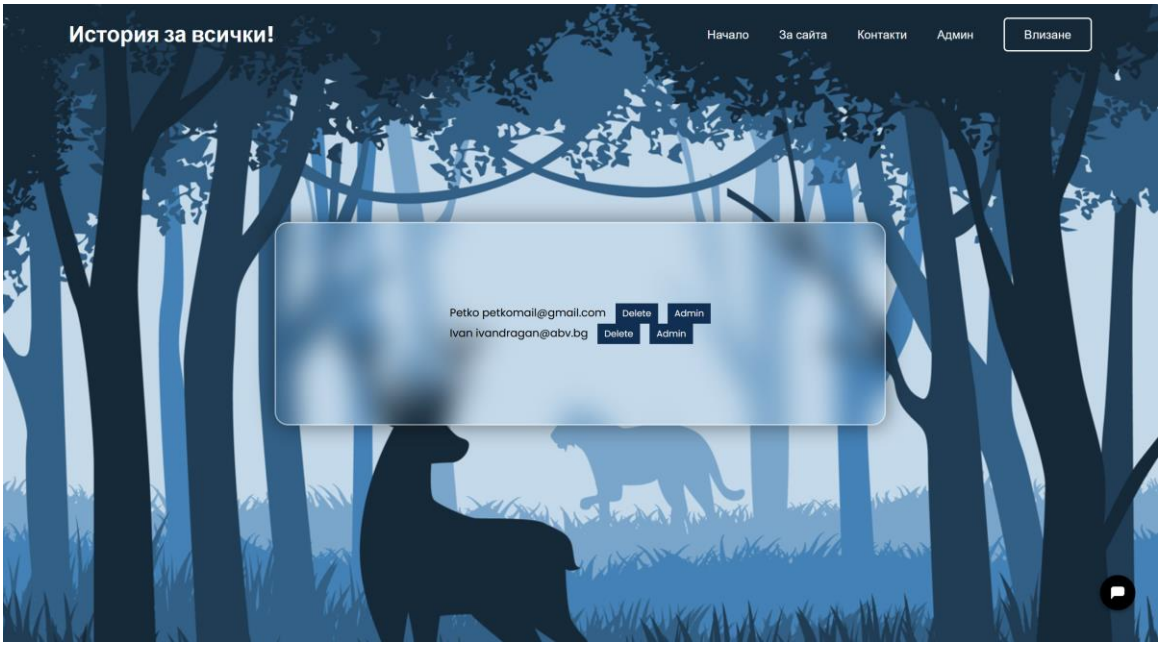
В Java кодът в следващата фигура, може да се види как директно се прави проверката за потребителят дали съществува като акаунт и съответно дали е админ. Информацията за потребителите се взима директно от базата данни и след това ако има съвпадение, се прави проверка дали и паролата, която в базата данни е хеширана, съвпада с дадената от потребителя парола за вход. Ако да, то ще върне успешна автентикация за вход.

```
1 usage
public boolean authenticateUser(String userName, String userPassword) {
    User user = userRepository.findByUsername(userName);

    return user != null && BCrypt.checkpw(userPassword, user.getPassword()); // User exists and password matches if not then authenticati
}

1 usage
public boolean adminCheck(String userName) {
    User user = userRepository.findByUsername(userName);

    return user != null && user.getIsAdmin(); // User exists and is admin or user doesn't exist or isn't admin
}
```

Фиг. 14 – Java код за проверка на вход на потребител и за админ

3.3. Даване на администрски права и триене на профили

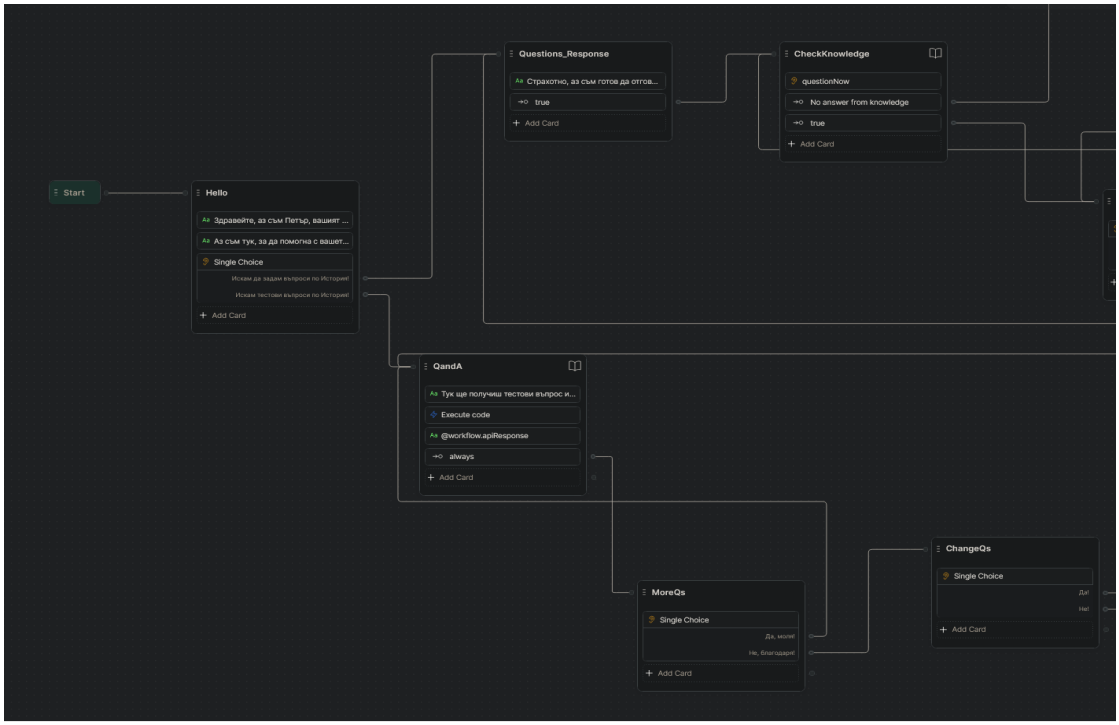
След като един админ отвори уеб приложението, ще може да влезе в частно меню, създадено специфично за администратори, където той може да трие потребителски профили или пък при общо взето решение, ще може да зададе администрски права на даденият потребител.



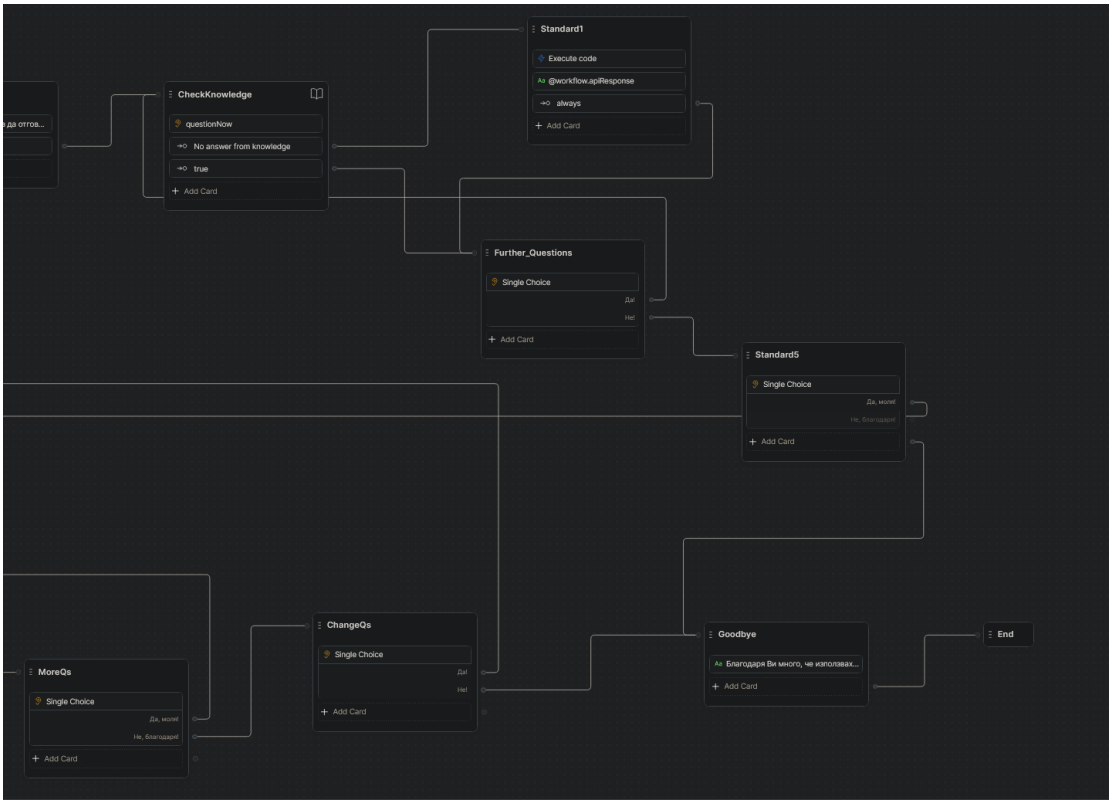
Фиг. 15 – Примерно администрско меню с бутони за действията

3.4. Интеракция с чат-бота

Опцията да се започне интеракция с чат-бота се намира на всяка страница. Чат ботът е създаден, чрез Botpress, което е open-source платформа за менажиране на чат ботове диалогови приложения. Там се дава възможността да се изгради самият чат бот, използвайки node-ове, които се навръзват един с друг и се задава текстови output на самия чатбот и какво да прави, когато получи конкретен отговор във вече подготвена ситуация. Самата платформа предоставя опростен вариант на изкуствен интелект, който използва база данни, която му давам предварително и отговаря на зададените му въпроси, спрямо тази база данни. Самата база данни включва няколко различни файлове, които съдържат възможно най-много информация за българската история. Тук се дават две възможности на потребителя: Първата е свободно да може да зададе въпроси относно историята и да получи отговори, които ботът е изградил от базата данни; Втората е ботът да дава вече готови тестови въпроси, където да може потребителят да тества своите знания.



Фиг. 16.1 – Първата половина от системата за диалог на чат бота



Фиг. 16.2 – Втората половина от системата за диалог на чат бота

Освен това частта от логиката на чат бота, в която се дава възможност свободно да се питат въпроси, за проверка и възможно най-точен и ясен отговор, се използва и StackAI, което работи на основата на ChatGPT, при което единствено му подавам въпросите като input, подавам му базата данни, която използвам и в Botpress и му давам откъс, който да му „обясни“ какво по-точно трябва да свърши с цялата тази информация и input.

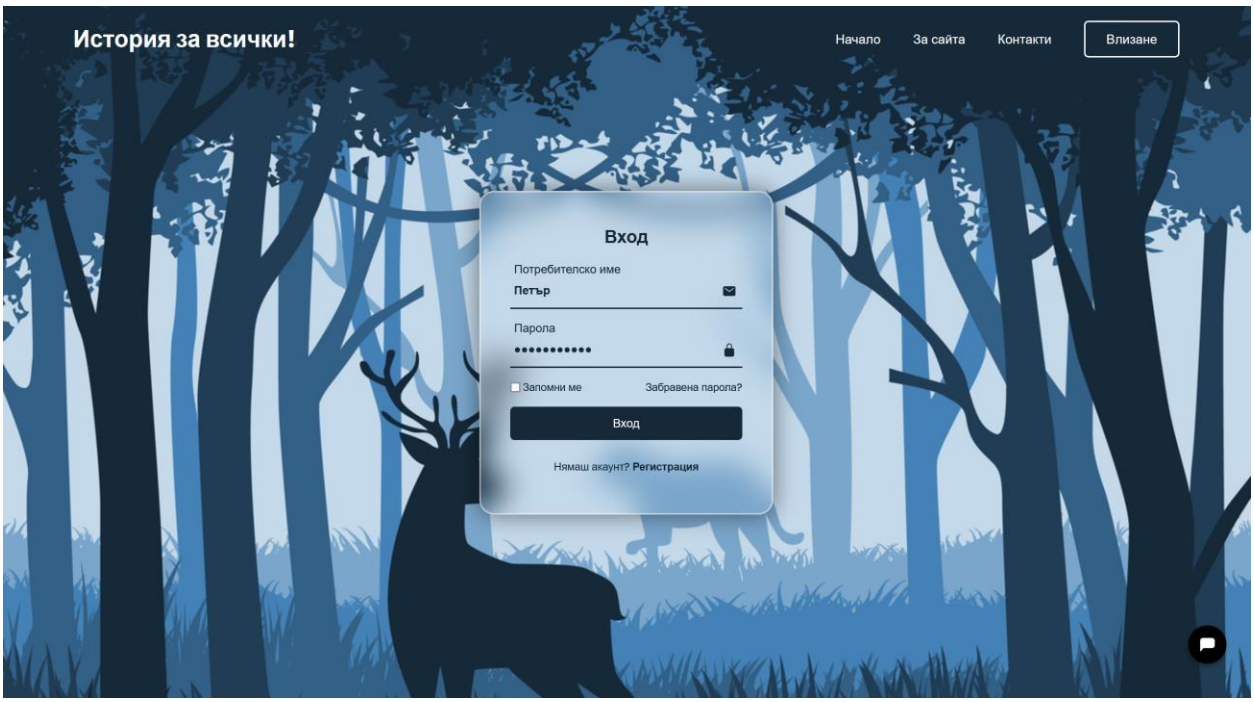
Фиг. 17.1/17.2 – Примерен диалог проведен с чатбота в уеб приложението

4. Изпълнение на нефункционалните изисквания

4.1.Сигурност

Сигурността е постигната по няколко начина:

Достъпване до допълнителната функционалност на приложението на чат-бота, след автентикация:



Фиг. 18 – Форма за вход в система със сигурност за паролата

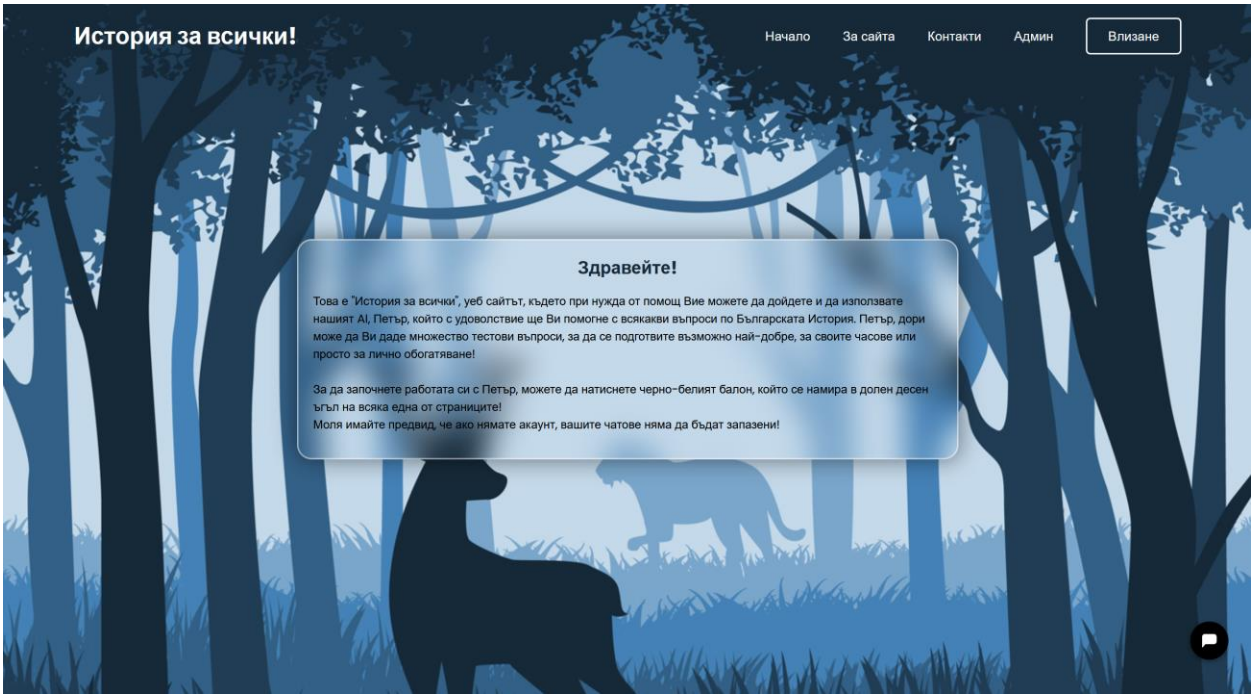
Съхраняване на паролите в хеширан вид. Хеширането се осъществява с помощта на библиотеката BCrypt.

Result Grid | Filter Rows: | Edit: | Export/Import:

Id	username	password	email	isAdmin
1	Peter	ZR"iyp0i	petar123@abv	1
2	Ivan	8/cnIrxd	vancho@abv.bg	1
3	Teodora	1jN9G"8S	tediti@gmail.com	0
4	xxPetre	h2Q&04JE	pepo43@gmail.com	0
5	Hitroncho	HYaI0&kr	hinoto@abv.bg	0
NULL	NULL	NULL	NULL	NULL

Фиг. 19 – База данни с хеширани пароли

И последно, все пак има администратори, съответно, за тях се появява нова страница, която им дава достъп до триене и даване на администрски права на други потребители.



Фиг. 20 – Начална страница, на която може да се види допълнният бутон за администрска страница

4.2. Разширяемост

Структурата на проекта следва MVC архитектурата, която разделя отговорностите по обработка на данните, визуализация и обработка на заявките, и по този начин улеснява

по-нататъшното развитие на приложението, тъй като лесно може да се добави нова функционалност в някой от трите компонента, без това да наложи промени останалите два.

4.3. Използваемост

Потребителският интерфейс е лесен за навигация и е разработен така, че да изисква минимални предварителни познания. Използвани са еднакви наименования и стилове за бутони, полета и други повтарящи се елементи в различните страници. Разработена е функционалност за предложения при попълване на текст.

Линк към GitHub хранилище

Кодът на разработеното приложение се намира в следното GitHub хранилище: https://github.com/AntonioDimitrov43/WebApplication-TUSofia

Заключение

С огромните темпове на развитие в модерното общество, ние продължаваме да натрупваме все повече и повече източници на информация и заедно с това и самото количество на тази информация се увеличава експоненциално. Но нека не забравяме, че в края на деня, ние сме хора, ограничени от физическите си тела и нямаме как да възприемем, опработим и запомним тези страшни количества информация.

Особено с това колко лесно достъпно е всичко в днешно време, има един фактор, който влияе на ученици и студенти все по-тежко. Това е натискът от техните преподаватели да знаят все повече, за по-малко количество време. Затова вярвам, че с този продукт, всички многократно биха увеличили темпото, с което получават обработена информация, но и също така количеството информация, която биха усвоили пълноценно.

Особено, за по-малки ученици, това би се оказало доста полезно средство, понеже би им създало чувството, че водят някакъв (дори то да бъде опростен) разговор, където не биха се срамували да зададат въпроси.

И като финал, като резултат от загрижеността към бъдещето ни развитие като мислещи хора, се породи тази идея за интерактивно Уеб приложение с цел подпомагане на обучението по история. И нека не забравяме, история е нашето минало и никога не трябва да я губим от спомените си, понеже ще повторим всички грешки на предците ни!

Използвана литература

1. https://botpress.com/
2. https://www.stack-ai.com/
3. https://spring.io/projects/spring-boot
4. https://www.java.com/en/download/help/whatis_java.html
5. https://dev.mysql.com/doc/
6. https://en.wikipedia.org/wiki/HTML
7. https://developer.mozilla.org/en-US/docs/Web/CSS
8. https://developer.mozilla.org/en-US/docs/Web/CSS
9. https://www.geeksforgeeks.org/hashing-in-java/
10. https://developer.mozilla.org/en-US/docs/Web/JavaScript
11. https://bg.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%91%D1%8A%D0%BB%D0%B3%D0%B0%D1%80%D0%B8%D1%8F
12. https://bg.wikipedia.org/wiki/%D0%9F%D1%8A%D1%80%D0%B2%D0%B0_%D0%B1%D1%8A%D0%BB%D0%B3%D0%B0%D1%80%D1%81%D0%BA%D0%B0_%D0%B4%D1%8A%D1%80%D0%B6%D0%B0%D0%B2%D0%B0
13. https://bg.wikipedia.org/wiki/%D0%92%D1%82%D0%BE%D1%80%D0%B0_%D0%B1%D1%8A%D0%BB%D0%B3%D0%B0%D1%80%D1%81%D0%BA%D0%B0_%D0%B4%D1%8A%D1%80%D0%B6%D0%B0%D0%B2%D0%B0
14. https://bg.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D1%82%D0%B0_%D0%B1%D1%8A%D0%BB%D0%B3%D0%B0%D1%80%D1%81%D0%BA%D0%B0_%D0%B4%D1%8A%D1%80%D0%B6%D0%B0%D0%B2%D0%B0
15. http://mvuiel.bg/wp-content/uploads/2019/06/%D0%98%D0%B7%D0%BA%D1%83%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%B8%D1%8F%D1%82-%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82.pdf
16. Applying Data Science: How to Create Value with Artificial Intelligence на Arthur Kordon, 2020г.
17. Artificial Intelligence: The Ultimate Guide to AI, The Internet of Things, Machine Learning, Deep Learning + a Comprehensive Guide to Robotics на Neil Witkins, 2020г.

ДЕКЛАРАЦИЯ
ЗА АВТОРСТВО НА ДИПЛОМНА РАБОТА

Долуподписаният/ата .....

(име, презиме, фамилия)

специалност : ..... фак. № .....

ДЕКЛАРИРАМ:

Представената от мен дипломна работа на тема: .....

.....

.....

е лична моя авторска разработка, резултат от собствени изследвания.

Потвърждавам, че тя в нейната цялост и отделни части не е била използвана за придобиване на образователна и/или научна степен в ТУ - София или в други университети.

Формулировки, идеи и текстове, взети от други източници, са цитирани точно и с коректно посочване на техните автори. Дипломната работа не е публикувана на друго място.

Декларирам, че предоставям правото на Факултет Приложна математика и информатика при ТУ - София, съгласно процедурите и правилниците на университета, да архивира и съхранява тази дипломна работа с цел доказване във времето на моето авторство.

Дата: ..... 202 ..... г. Дипломант:

София