МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

САТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
доктор Ph.D
_____М.Тұрдалыұлы
«__06__» ____июня_____ 2021 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: " «Cookıng book» - электронная кулинарная книга."

по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнила Спирикова Д.З.

Научный руководитель
Доктор Ph.D, зам. директора
_____Ж.М.Алибиева
«__06__» ____июня_____ 2021 г.

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
Доктор Ph.D
_____М. Тұрдалыұлы
«_06_» _____июня_____ 2021 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся Спириковой Диане Зафаровне
Тема: «Cookıng book» - электронная кулинарная книга.
Утверждено приказом ректора №2131-б «24» ноября 2020 г
Дата сдачи проекта «08» июня 2021 г
Исходные данные к дипломному проекту: Описание необходимых функций проекта.
Перечень подлежащих разработке в дипломном проекте вопросов:
а) реализация CRUD системы;

б) реализация защищенной авторизации, регистрации и аутентификации;
в) проектирование и разработка пользовательского интерфейса;

г)разработка разделения пользователей на роли с соответствующими возможностями;
Перечень графического материала (с точным указанием обязательных чертежей): представлены 20 слайдами презентации.

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
1. Анализ предметной области, разработка технического задания	08.04.21	Выполнено
2. Выбор базового компонента	14.04.21	Выполнено
3. Разработка дизайна интерфейса	16.04.21	Выполнено
4. Разработка функционала для серверной части	20.04.21	Выполнено
5. Разработка функционала фронтенд стороны	10.05.21	Выполнено
6. Тестирование приложения	15.05.21	Выполнено
7. Написание пояснительной записки к дипломному проекту	19.05.21	Выполнено

Подписи

консультантов и нормоконтролера на законченный дипломный проект с указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Программное обеспечение	Ассистент Рамазан А. Б.	31.05.2021	
Нормоконтролер	Магистр технических наук, лектор Марғұлан Қ.	06.06.2021	

Научный руководитель _____ Алибиева Ж. М.

Задание принял к исполнению обучающийся _____ Спирикова Д.З.

Дата «6 » июня 2021 г.

АННОТАЦИЯ

Этот дипломный проект посвящен разработке динамического сайта в формате SPA. SPA (Single Page Application) – веб-приложение, которое вместо разных страниц подгружает нужные элементы из отдельных компонентов, это помогает увеличить скорость работы сайта и уменьшить его вес.

Написанный сайт позволит пользователям делиться рецептами на единой площадке.

Для реализации должен использоваться JavaScript фреймворк NestJs для серверной части и фреймворк Angular для браузерной стороны, с языком программирования TypeScript. База данных для хранения записей – MongoDb, облачный нереляционный кластер, который обеспечивает бесплатным облачным хранилищем, хорошо защищенным и оптимизированным.

Данное техническое задание состоит из таких разделов как общие сведения, назначение, цель, функциональная характеристика проекта, теоретическая часть.

АҢДАТПА

Бұл дипломдық жоба SPA форматындағы динамикалық веб-сайтты жасауға арналған. SPA (Single Page Application) - бұл әртүрлі беттердің орнына қажетті элементтерді бөлек компоненттерден жүктейтін веб-қосымшаның түрі, ол сайттың жылдамдығын арттыруға және оның салмағын азайтуға көмектеседі.

Жасалған сайт қолданушыларға рецептілерді бір сайтта бөлісуге мүмкіндік береді.

Іске асыру үшін JavaScript NestJs фреймворкі серверге, ал бұрыштық браузерге TypeScript бағдарламалау тілінде қолданылуы керек. Жазбаларды сақтауға арналған мәліметтер базасы - бұл бұлтқа негізделген, тегін қауіпсіз бұлт қоймасы бар, бейреляционды кластер.

Бұл техникалық тапсырма жалпы ақпарат, мақсат, қолдану аясы, жобаның функционалдық сипаттамалары, теориялық бөлім деген бөлімдерден тұрады.

ANNOTAION

This thesis project is dedicated to the development of a dynamic website in the SPA format. SPA (Single Page Application) is a web application that loads the necessary elements from separate components instead of different pages. This helps to increase the speed of the site and reduce its weight.

The site will allow users to share recipes on a single platform.

For the implementation should use the JavaScript framework NestJs for the server-side and the Angular framework for the browser side, with the TypeScript programming language. For storing records in the database we should use MongoDb, a cloud-based non-relational cluster that provides free cloud storage that is highly secure and optimized.

This technical assignment consists of such sections as general information, purpose, purpose, functional characteristics of the project, theoretical part.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На данный момент в интернете много сайтов, которые предоставляют пользователям возможность добавлять и просматривать рецепты, но, к сожалению, согласно статистике, пользователи не часто ищут рецепт именно на сайте, а именно «гуглят» запрос и кликают на первую ссылку. Хотелось бы понять, почему именно пользователям удобно не придерживаться определенного сайта, а использовать информацию непосредственно из поисковика.

Данная работа направлена на изучении uxui составляющей и попытке понять, почему именно пользователям не удобно брать информацию с сайта, проанализировать ошибки подобных сайтов и постараться их исправить в дипломном проекте.

1 Исследовательский раздел

1.1 Цель разработки

Цель данного проекта дать возможность пользователям создавать, изменять, удалять и хранить собственные посты с рецептами в базе данных, а также регистрировать, распределять роли и сохранять данные пользователей в отдельную таблицу в базе данных. Данные, которые будут сохранены в базе данных постов: название поста, описание поста, прикрепленная фотография или фотографии, автор поста, время создания. В базе данных пользователя сохраняются эти данные: имя, никнейм, электронная почта, зашифрованный пароль, роль пользователя (админ, модератор, юзер), фотография пользователя (аватарка), связь с постами, которые делал пользователь.

Все элементы и данные должны отслеживать изменения в базе данных и обновляться автоматически.

В техническом плане достоинство данного сайта состоит в быстроте работы, в визуальном плане это удобство дизайна и интерфейса пользователя, так называемом UxUi.

1.2 Определения, термины и сокращения

В таблице 1 предоставлены все термины и сокращения

Сокращение или термин	Определение
БД	База данных
СУБД	Система управления базами данных
API	Application Programming Interface
ПО	Программное обеспечение
Visual Studio Code	Редактор исходного кода
TS	TypeScript – язык программирования, расширяющий возможности JavaScript
HTML	Hypertext Markup Language. Язык гипртекстовой разметки.
UX	User Experience – опыт пользователя
Фреймворк	Программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Таблица 1 – Сокращения, термины и их определения

2 Технологический раздел

2.1 Фреймворк Angular

Angular[1] это фреймворк от компании Google, который был разработан для создания клиентских приложений. Первоначальная его цель была предоставить пользователям средства для разработки SPA-приложений[2] – одностраничных приложений. Angular часто путают с AngularJs[3]. Хотя они и похожи, Angular был разработан несколько позже и позиционируется как принципиально новый фреймфорк.

Благодаря Angular можно создавать MVP[4] подобные приложения с динамическим изменением передаваемых параметров. Приложение будет работать на основе компонентов и сервисов, которые выполняют основную логику приложения.

Сам фреймворк использует язык typescript для создания логики, что является для него одним из особенностей.

2.1.1 Среда разработки

Среда разработки – это специализированное программное обеспечение, которое устанавливается на ваш компьютер. Были разработаны для упрощения процесса разработки приложений.

Из самых популярных можно рассмотреть такие среды разработки, как:

Visual Studio Code – бесплатный, имеющий огромную популярность тестовой редактор исходного кода от компании Microsoft. Совместим с такими операционными системами, как Windows, macOs, Linux. Из основных плюсов данный текстовой редактор занимает мало места, отлично подходит для кроссплатформенной разработке приложений, позволяет дополнительно докачивать в него дополнения, которые улучшают как и качество кода, так и ускоряют разработку приложения.

Notepad++ - также бесплатный текстовой редактор, код которого является открытым. Поддерживается только для операционной системы Windows. Из основных плюсов поддерживает такие языки, как Verilog VHDL. Хот дизайн является достаточно устаревшим, все еще занимает свое место на поприще текстовых редакторов.

Sublime Text – текстовой редактор с ограниченной бесплатной лицензией. Имеет приятный дизайн интерфейса и хороший функционал, но больше заточен для разработки небольших сайтов, или сайтов-визиток.

Оценив и рассмотрев все вышеперечисленные редакторы выбор был сделан в сторону Visual Studio Code

2.2 NestJs

2.2.1 Преимущества NestJs

NestJS[5] используется для написания масштабируемых, тестируемых и слабосвязанных приложений. Основанный на Node.js, ун приносит масштабируемый Node.js на совершенно новый уровень. Фреймворк поддерживает такие базы данных, как PostgreSQL, MongoDB, MySQL[6].

В среде разработчиков говорят, что NestJs – это фреймворк, созданный для облегчения жизни и работы разработчика. С его помощью можно использовать правильные и нужные архитектурные подходы в программировании. Созданный недавно, он считается самым перспективным фреймворком для бэкенда на данный момент. К тому же, его архитектура очень похожа на архитектуру Angular, что тоже упрощает разработку программистам, уже знакомым с вышеупомянутым фреймворком для фронтенда.

Благодаря NestJs фреймворку в разработке можно использовать лучшие передовые концепции программирования, как DDD(Domain Driven Design)[7], Event sourcing[8], а также микросервисную архитектуру[9].

NestJs фреймворк, написанный на TypeSctipt, очень легко подходит для текстов, и содержит все необходимое для разработки. При его созданbи разработчики вдохновлялись Angular, поэтому стилистика и язык можно сказать одинаковые.

2.2.2 Функции NestJs

Как уже говорилось выше, авторы фреймворка вдохновлялись структурой Angular, поэтому NestJs получился достаточно походим на него. Все рабочие модули можно как создать вручную, так и создать с помощью специального инструмента, который создаст нужный компонент и впишет его в основную структуру проекта[10].

Контроллеры – controllers. Данный слой ответственен за обработку запросов, а также возвращают ответ клиенту. В сгенерированном или созданном файле контроллера за его идентификацию как контроллера отвечает декоратор @Controller(), который импортируется из общей библиотеки @nestjs/common[11].

Провайдеры – providers. К ним относятся сервисы(service), репозитории(repository), factory и helper и тд. Они могут быть внедрены как в контроллеры, так и в другие провайдеры. За провайдеры отвечает декоратор @Injectables(), который также импортируется из общей библиотеки @nestjs/common.

Модули – modules. Поставляется декоратором @Module(). Он предоставляет данные (метаданные), которые фреймворк использует для построения структуры приложения. При создании нового проекта он уже имеет корневой модуль. В маленьких приложениях одного модуля будет достаточно, но в больших точно будет использоваться несколько, для удобства и оптимизации кода. Также импортируется из @nestjs/common.

Также через модули идет подключение к базе данных, загрузка новых сервисов и контроллеров.

3 Проектная часть

3.1 Архитектура проекта

Приложение будет разработано на основе такого стека технологий[12], как NestJs, Angular и MongoDb, как показано на рисунке 3.1.

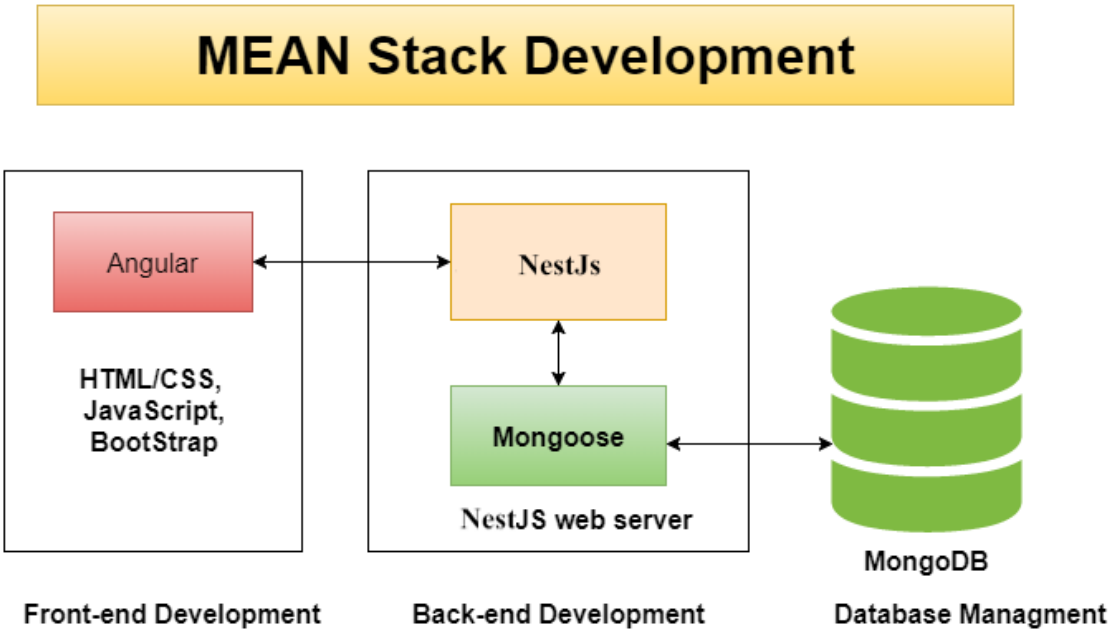

Рисунок 3.1 – Стек технологий для разработки приложения

Данные будут храниться в нереляционной базе данных MongoDb[13] в качестве документов. Через фреймворк NestJs мы получим возможность получать, редактировать и всячески изменять данные из базы данных. Данные поступят на контроллер, и с помощью технологий Mongoose будут обработаны в связи с нашими нуждами и распределены в нужные модули по запросу. После определенных манипуляций данные смогут поступить на внешний уровень, уровень фронтенда. У каждого компонента в Angular имеется слой с логикой, именно он и будет запрашивать нужные нам данные. Сама логика будет прописана в сервисах, а компонент будет иметь доступ с этим сервисам[14]. Далее, когда наши компоненты получили данные, они их смогут обработать, если в этом есть необходимость и отправить на html слой, слой, который отвечает на интерфейс пользователя. И именно там данные смогут проявиться в более удобном для пользователя виде.

3.2 Описание диаграммы деятельности

Диаграмма деятельности – или диаграмма активности, это UML диаграмма. Она представляет последовательность действий, аналогична диаграмме потока данных. UML – это унифицированный язык моделирования (Unified Modeling Language), система обозначений, которую применяют для объектно-ориентированного анализа. Говоря простым языком, это схемы[15]. Плюсы построения таких диаграмм:

- позволяет посмотреть задачу с различных точек зрения;
- другим разработчикам становится легче понять суть задачи и ее реализацию;
- такие диаграммы позволяют быстро ознакомиться с синтаксисом приложения;

Действия в данной диаграммы можно показывать как последовательными, так и параллельными, но обязательно она будет иметь начальное и конечное состояние.

Данный проект стартует с того, что пользователь находится на главной странице. Если пользователь хочет воспользоваться функционалом сайта, то пользователю предлагают авторизацию. Если пользователь не авторизован на сайте, то его перенаправят на страницу регистрации, после которой будет перенаправление на страницу авторизации в приложение. Если пользователь не зарегистрирован или не авторизован, у него также будет возможность пользоваться приложением, однако с ограниченным функционалом.

После регистрации и/или авторизации в приложении с технической стороны будут происходить изменения и запросы на серверную часть проекта. В итоге, при успешном запросе серверная часть вернет токен, принадлежавший пользователю. Таким образом аутентификация будет пройдена. С помощью аутентификации и токена у пользователя появится расширенный функционал пользования сайтом.

Пользователя перенаправят на главную страницу сайта, через которую он сможет воспользоваться нужным ему функционалом, таким как фильтрация и поиск рецептов, добавление поста в избранное, создание нового рецепта, просмотр собственного профиля, настройка профиля, доступ к избранным рецептам. В зависимости от роли пользователя у него в личном профиле может быть возможность перехода в состояние администратора, у которого расширенный список функционала. Администратор может модерировать, удалять рецепты, просматривать расширенную информацию про пользователя, назначать роли.

Чтобы создать подобную диаграмму были использованы готовые решения. С помощью инструмента «draw io»[16] можно создавать диаграммы разной сложности и ветвленности. На рисунке 3.2 представлена диаграмма деятельности проекта.

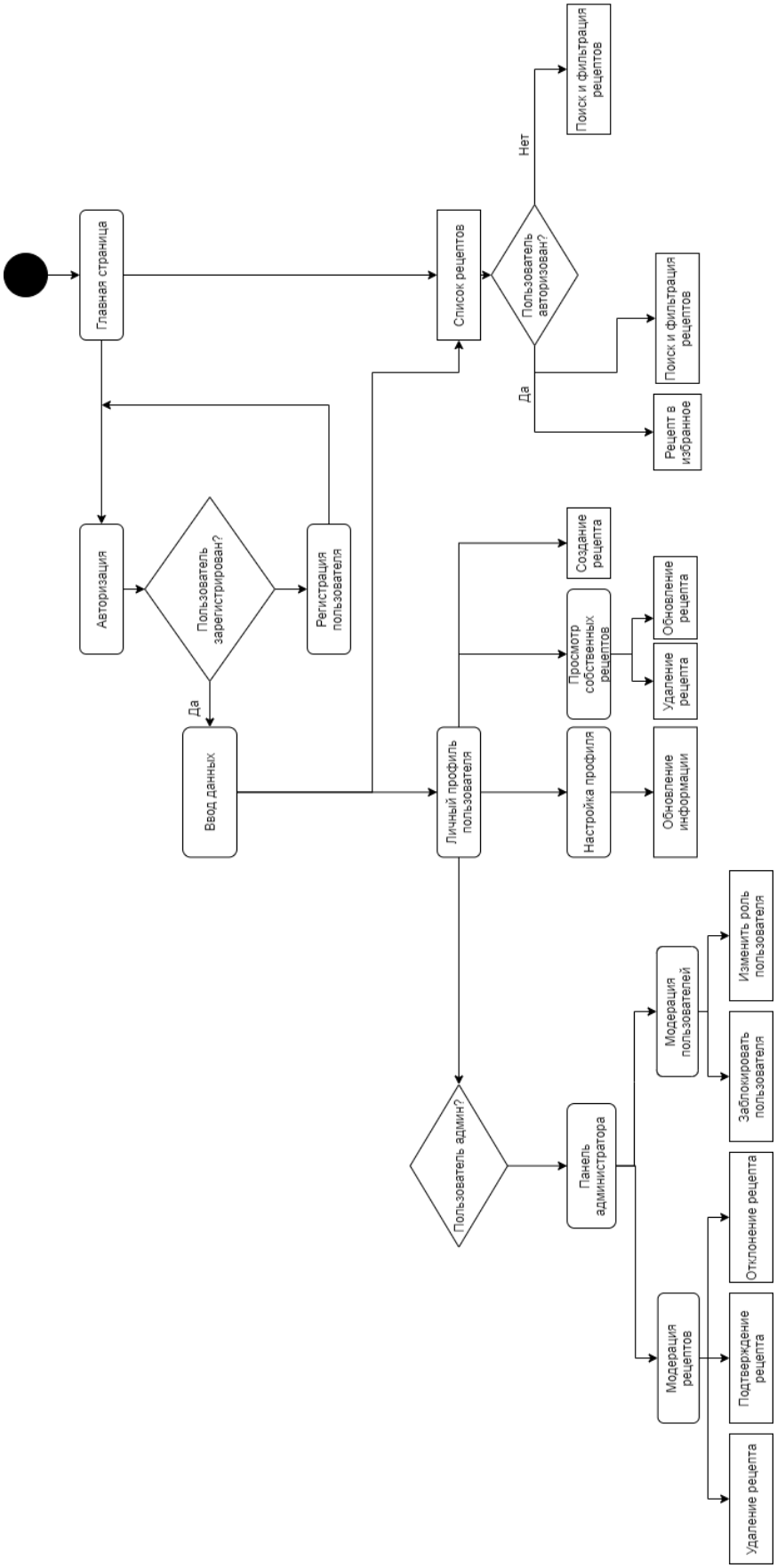

Рисунок 3.2 – диаграмма деятельности проекта

3.3 Разработка серверной части

Серверная часть проекта, или бэкенд – главная составляющая любого современного динамического приложения. Вся логика, работа с данными и любые другие манипуляции происходят на серверной стороне.

Работа с бэкендом будет вестись во фреймворке NestJs, с концепцией MVP. Данные будут поступать в контроллер, далее передаваться в сервисы. После этого произойдёт подключение к моделям и схемам, которые дают доступ к базе данных. В зависимости от ответа базы, данные будут возвращены или нет.

3.3.1 Аутентификация и авторизация

NestJs имеет отличную совместимость с библиотекой JWT[17], которая предоставляет возможность создавать, модифицировать, шифровать, отправлять и удалять из браузера токен[18]. Токен создается в момент, когда пользователь регистрируется/авторизуется на сайте и хранит в себе данные о пользователе. Это позволяет проверять, авторизован ли пользователь на протяжении всей рабочей сессии. Все данные шифруются с помощью секретного слова и алгоритма шифрования, который предоставляется библиотекой JWT. Сам токен состоит из нескольких частей, которые отвечают за определенные части данных. Ниже, на рисунке 3.3 предоставлен пример зашифрованного и расшифрованного токена с помощью сайта jwt.io.

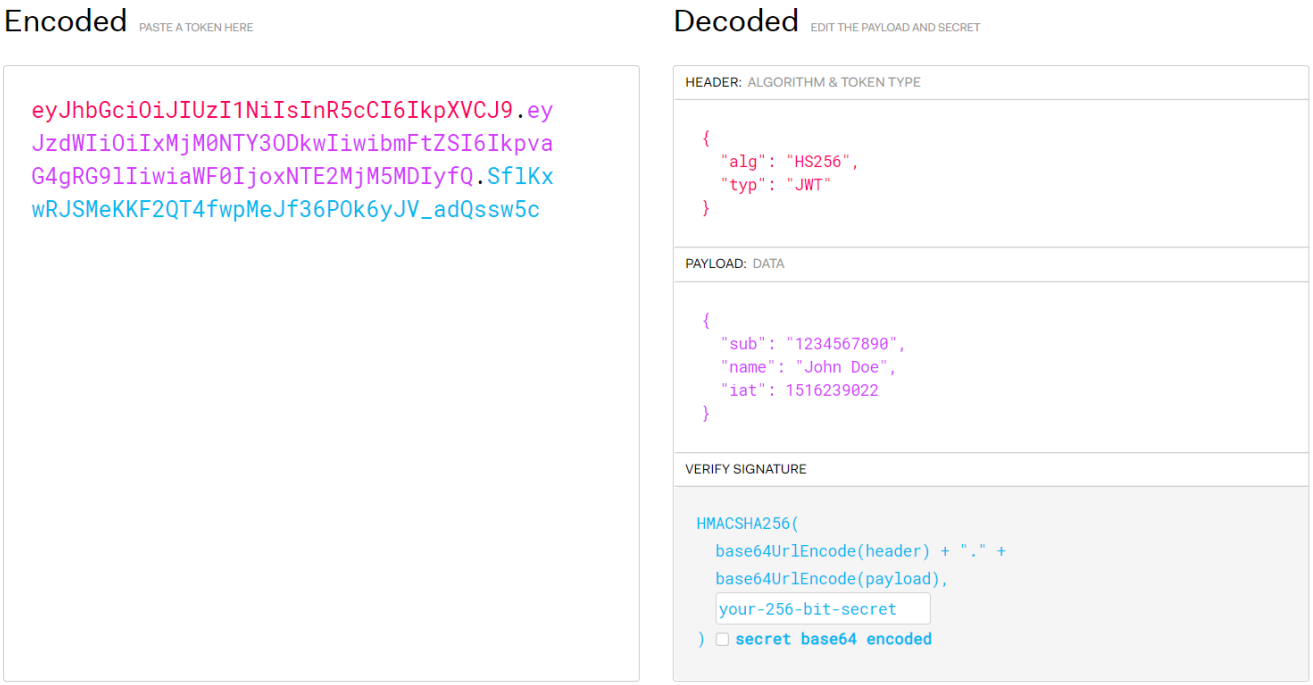

Рисунок 3.3 – пример зашифрованного и расшифрованного токена

Настройки конфигурации jwt токена на api стороне выглядят так, как показано на рисунке 3.4. Регистрация токена происходит в асинхронном формате, что позволяет нам предоставлять пользователю его токен за считанные миллисекунды, все зависимости от того, сколько еще одновременно пользователей совершают авторизацию. Секретная фраза и в строке 17 была вынесена в отдельный .env[19] файл для большей безопасности и красоты разделения кода.

```
1  import { forwardRef, Module } from '@nestjs/common';
2  import { ConfigModule, ConfigService } from '@nestjs/config';
3  import { JwtModule } from '@nestjs/jwt';
4  import { UserModule } from 'src/user/user.module';
5  import { JwtAuthGuard } from './guards/jwt-guard';
6  import { JwtStrategy } from './guards/jwt-strategy';
7  import { RolesGuard } from './guards/roles.guard';
8  import { AuthService } from './services/auth.service';
9
10 @Module({
11     imports: [
12         forwardRef(() => UserModule),
13         JwtModule.registerAsync({
14             imports: [ConfigModule],
15             inject: [ConfigService],
16             useFactory: async (configService: ConfigService) => ({
17                 secret: configService.get(`JWT_SECRET`)
18             })
19         })
20     ],
21     providers: [AuthService, RolesGuard, JwtAuthGuard, JwtStrategy],
22     exports: [AuthService]
23 })
24 export class AuthModule {}
25
```

Рисунок 3.4 – написание auth модуля и добавление jwt токена

После успешной операции по добавлению пользователю токена, мы сможем пронаблюдать его в браузере. На рисунке 3.5 показана строка токена, который хранится в локальном хранилище. Если пользователь будет выходить из своего аккаунта, то данный токен будет удален из локального хранилища.

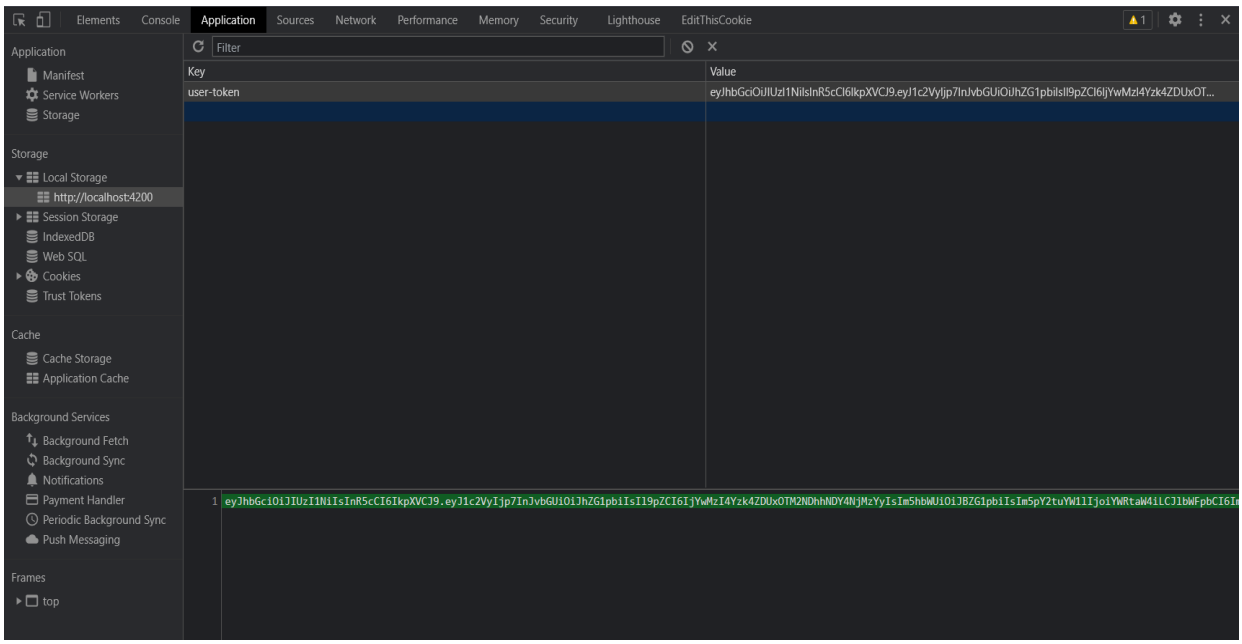

Рисунок 3.5 – токен в локальном хранилище браузера

3.3.2 База данных MongoDb

База данных MongoDb – нереляционная база данных, где данные представляются как документы. Она не требует описания схемы таблиц. Отличный пример no-Sql системы[20], и использует JSON и только описанную схему (schema). Очень удобна для построения сложносвязанных систем, где много вложенных друг в друга слоев документов, поэтому при выборе базу данных выпал пал именно на нее. Еще одно ее достоинство – это то, что она является облачной. Это облачный кластер, который можно как приобрести за валюту, так и получить бесплатно, в зависимости от ваших нужд. Разница составляет объём предоставляемой памяти.

Для работы с данной базой в NestJs нужно установить библиотеку mongoose, которая предоставляет расширенные инструменты работы с базой. Далее в коде прописывается строчка кода, которая ведет на наш созданный облачный кластер. Для того, чтобы в базе создать разные таблицы, нужно прописывать файл схемы.

Ниже на рисунке 3.6 показан пример схемы пользователя сайта.

Рисунок 3.6 – схема пользователя сайта

3.4 Разработка UXUI

Разработка UXUI[21] составляющей велась во фреймворке Angular. Он поддерживает большое количество uxui решений, таких как materal, angular material и bootstrap.

3.4.1 Angular Material и Bootstrap

Angular Material[22] – библиотека, которая предоставляет доступ к уже созданным и настроенным компонентам Angular.

Bootstrap[23] – набор инструментов, отличный помощник для создания адаптивных сайтов.

Оба этих инструмента были использованы в разработке. Angular Material дает возможность пользоваться уже настроенными красивыми компонентами, которые соответствуют правилам UI[24], имеют минимальную настройку интерфейса и лаконичный дизайн. Bootstrap же в проекте использовался для

построения сетки макета сайта, распределении блока компонентов и минимальной адаптации. Хочу заметить, что адаптируемости сайта не было предпочтения.

Чтобы пользоваться компонентами Angular Material нужно установить основную библиотеку, затем отдельно импортировать каждый, нужный вам компонент. Пример импортированных компонентов предоставлен на рисунке 3.7.

```
],
imports: [
  CommonModule,
  // BrowserModule,
  AppRoutingModule,
  BrowserAnimationsModule,
  MatToolbarModule,
  HttpClientModule,
  FormsModule,
  ReactiveFormsModule,
  MatAutocompleteModule,
  MatExpansionModule,
  MatFormFieldModule,
  MatInputModule,
  MatButtonModule,
  MatSelectModule,
  MatTableModule,
  MatPaginatorModule,
  MatCardModule,
  MatChipsModule,
  MatProgressBarModule,
  MatMenuModule,
  MatProgressSpinnerModule,
  MatSnackBarModule,
  MatIconModule,
  MatSidenavModule,
  MatDialogModule,
  MatTabsModule,
  MatTooltipModule,
  MatStepperModule,
  MarkdownModule.forRoot()
],
```

Рисунок 3.7 – пример импортированных компонентов Angular Material

Для того, чтобы использовать Bootstrap нужно прописать строчку кода в глобальных стилях. Чтобы применить настройки bootstrap к любому компоненту нужно указать специальный класс. Например, как указано в рисунке 3.8.

```
<div class="row row-div">
    <div class="col" *ngIf="postSource">
        <mat-card>
            <div class="row" style="margin: 10% 5% 5% 5%;">
                <div class="col">
                    <h1>{{postSource.totalDocs}}</h1>
                    <p>Всего постов</p>
                </div>
                <div class="col-2">
                    <mat-icon>home</mat-icon>
                </div>
            </div>
        </mat-card>
    </div>
```

Рисунок 3.8 – использование классов Bootstrap на компонентах

ЗАКЛЮЧЕНИЕ

Результатом данной дипломной работы является многокомпонентное одностраничное SPA приложение « Cooking book », которое позволяет пользователям создавать, просматривать рецепты и многое другое. Данное приложение ориентировано на общую аудиторию и является свободным и распространяемым ресурсом. Основными требованиями по созданию приложения были функциональные возможности, какие как:

- разделение на роли
- CRUD система, а именно:
    - создание рецепта
    - редактирование рецепта его автором
    - удаление рецепта его автором
    - обновление рецепта его автором
    - создание личного профиля пользователя
    - обновление личных данных в профиле пользователя
- удаление (блокировка) пользователя другим пользователем, с ролью администратора
- использование Guards
- защищенная авторизация и аутентификация

В процессе разработки приложение обрело собственный стиль, дополнилось функционалом, которые запросы были пересмотрены. Во время разработки были открыты новые возможности фреймворков NestJs и Angular, приняты во внимание многие современные методологии и концепции программирования. Произошло знакомство с облачной нереляционной базой данных MongoDb, которая значительно улучшила качество разработки и скорость работы самого приложения.

Проект был создан с помощью фреймворков NestJs на языке TypeScript, Angular и базы данной MongoDb, которая является noSql системой. Все вышеперечисленные цели и требования были достигнуты.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Документация Angular // Электронная версия на сайте https://cli.angular.io/
2. Что такое SPA приложения // Электронная версия на сайте https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya
3. Angular и AngularJs – в чем разница? // Электронная версия на сайте https://web-shpargalka.ru/angular-i-angularjs-v-chem-raznica.php
4. Почему вашему стартапу нужен MVP? // Электронная версия на сайте https://stfalcon.com/ru/blog/post/why-your-startup-needs-mvp
5. Документация NestJs // Электронная версия на сайте https://docs.nestjs.com/
6. Современные базы данных // Электронная версия на сайте https://proglib.io/p/11-tipov-sovremennyh-baz-dannyh-kratkie-opisaniya-shemy-i-primery-bd-2020-01-07
7. DDD // Электронная версия на сайте https://habr.com/ru/post/334126/
8. Знакомимся с Event Soursing // Электронная версия на сайте https://habr.com/ru/company/otus/blog/518282
9. Что такое микросервисная архитектура // Электронная версия на сайте https://mcs.mail.ru/blog/prostym-jazykom-o-mikroservisnoj-arhitekture
10. Введение в структуру проекта NestJs // Электронная версия на сайте https://coderlessons.com/articles/veb-razrabotka-articles/vvedenie-v-nest-js-dlia-razrabotchikov-angular
11. Библиотека @nestjs/common // Электронная версия на сайте https://www.npmjs.com/package/@nestjs/common
12. Стек MEAN // Электронная версия на сайте https://itproger.com/course/mean
13. Документация MongoDb // Электронная версия на сайте https://www.mongodb.com/
14. Connecting NestJs with Angular // Электронная версия на сайте https://www.codemag.com/Article/2005051/NestJS-Step-by-Step-Connecting-NestJS-with-Angular-Part-4
15. UML диаграммы // Электронная версия на сайте https://evergreens.com.ua/ru/articles/uml-diagrams.html
16. Online tool for diagrams https://app.diagrams.net/
17. JSON Web Tokens // Электронная версия на сайте https://jwt.io/
18. Токен авторизации // Электронная версия на сайте https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc
19. Файл env // Электронная версия на сайте https://phpprofi.ru/blogs/post/72

20. NoSql системы // Электронная версия на сайте https://habr.com/ru/post/152477/
21. UXUI differences // Электронная версия на сайте https://www.usertesting.com/blog/ui-vs-ux
22. Angular Material UI component library // Электронная версия на сайте https://material.angular.io/
23. Bootstrap // Электронная версия на сайте https://getbootstrap.com/
24. Правила UI // Электронная версия на сайте https://ux.pub/47-vazhnyh-sovetov-dlya-ui-i-ux-dizaynerov/

Приложение А
(обязательное)

Техническое задание

А.1 Техническое задание на разработку динамического SPA приложения «COOK BOOK»

Настоящее техническое задание распространяется на разработку динамической SPA системы « Cookıng book », предназначеной для предоставления возможности пользователям создавать, изменять, удалять и хнарить собственные посты с рецептами в базе данных, а также регистрировать, распределять роли и сохранять данные пользователей в отдельную таблицу в базе данных.

А.1.1 Основание для разработки

Приложение разрабатывается на основе устного согласия научного руководителя выбранной дипломником темой.

А.1.2 Назначение

Система предназначена для обработки, хранения информации, созданной и предоставленной пользователями данных, связанных с кулинарными рецептами.

А.1.3 Требования к функциональным характеристикам

С помощью JavaScript фреймворка NestJs и Angular нужно создать сайт с возможностями данных функций:

Показать посты, отфильтрованные по тематике на главной странице сайта. При нажатии на карточку отдельного рецепта нужно показать всю дополнительную информацию, такую как название рецепта, описание, фотография/фотографии, дата создания.

Продолжение приложения А

Чтобы иметь возможность добавлять, обновлять, удалять посты, т.е, взаимодействовать с сайтом пользователю нужно зарегистрироваться. Для этого нужно реализовать функции:

– Регистрация: запрос у пользователя таких данных как имя, никнейм, электронная почта, пароль, повторный ввод пароля для безопасности. После удачной регистрации пользователя нужно перенаправить на страницу входа в профиль.

– Вход в профиль: запрос у пользователя таких данных как электронная почта и пароль. После удачного входа в профиль пользователю должен быть выдан зашифрованный токен, содержащий в себе объект введенных данных и роль.

После входа в профиль у пользователя появится возможность взаимодействовать с сайтом. Для этого понадобятся такие функции как:

– Создание поста: добавляет в базу введенные пользователем данные – название рецепта, фотография/фотографии, описание, автор. Дата создания поста добавляется автоматически.

– Обновление поста: будет обновляться уже созданный пост в базе данных данными, которые ввел пользователь. Обновить можно все, кроме автора и даты создания. Пользователь может обновлять только созданные им посты.

– Удаление поста: из базы удаляется пост. Пользователь может удалить только созданные им посты.

– У пользователя в профиле должны быть настройки и возможность обновления данных:

– Добавить/обновить/удалить фотографию: добавляет/удаляет/обновляет фотографию пользователя в его профиле. Пользователь может взаимодействовать только с его аватаркой.

– Сменить пароль: позволяет пользователю изменить его пароль. Пользователь может взаимодействовать только с собственными данными.

У зарегистрированных пользователей присвоены роли, которые разделяются на: User, Moderator и Admin

User – зарегистрированный пользователь сайта, у которого есть права:

– иметь личный кабинет/личную страницу, в которой содержатся персональные данные;

– иметь возможность просматривать рецепты на сайте без какие-либо ограничений;

– иметь возможность добавлять рецепт в базу данных рецептов;

– иметь возможность редактировать либо удалить рецепт собственного авторства;

– иметь возможность добавить рецепт в избранное для дальнейшего просмотра в отдельной вкладке «Избранное»;

Admin – зарегистрированный пользователь сайта, у которого есть права:

– тот же самый список прав, что и у пользователя User;

– имеется вкладка «Модерирование», где предоставлен список рецептов, которые еще не прошли модерацию;

– возможность выдавать/изменять другим пользователям тип роли;

А.1.4 Требования к надежности

Методы в контролере, относящиеся к роли «Админ», должны быть защищены Guard, для предотвращения несанкционированного доступа к этому разделу пользователями, не относящимися с этой роли.

Методы в контролере, относящиеся к удалению, обновлению и созданию рецепта должны быть защищены Guard, который проверяет, что пользователь является автором данных рецептов, для предотвращения несанкционированного доступа к данным чужими пользователями, которые не являются авторами рецепта.

Методы в контролере, относящиеся к удалению, обновлению данных пользователя должны быть защищены Guard, который проверяет, что пользователь является владельцем своих данных, для предотвращения несанкционированного доступа к данным чужими пользователями, которые не являются носителями этих данных.

Строка с секретной фразой, данные подключения к базе данных, логин и пароль подключения к базе данных должны быть вынесены в отдельный защищенный .env файл, для предотвращения несанкционированного доступа к базу данных.

Приложение Б
(обязательное)

Текст программы

— Код из файла app.module.ts

```
import { APP_INITIALIZER, NgModule } from '@angular/core';
// import { CommonModule  } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatToolbarModule } from '@angular/material/toolbar';
import { LoginComponent } from './components/login/login.component';
import { RegisterComponent } from './components/register/register.component';
import { HttpClientModule, HTTP_INTERCEPTORS } from
'@angular/common/http';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatButtonModule } from '@angular/material/button';
import { MatSelectModule } from '@angular/material/select';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatCardModule } from '@angular/material/card';
import { JwtHelperService, JWT_OPTIONS } from '@auth0/angular-jwt';
import { JwtInterceptor } from './interceptors/jwt.interceptor';
import { MatProgressBarModule } from '@angular/material/progress-bar';
import { MatIconModule } from '@angular/material/icon';
import { HomeComponent } from './components/home/home.component';
import { MarkdownModule } from 'ngx-markdown';
import { UsersComponent } from './components/user/users/users.component';
import { UserProfileComponent } from './components/user/user-profile/user-
profile.component';
import { UpdateUserProfileComponent } from './components/user/update-user-
profile/update-user-profile.component';
import { AllBlogEntriesComponent } from './components/blog/all-blog-entries/all-
blog-entries.component';
import { CreateBlogEntryComponent } from './components/blog/create-blog-
entry/create-blog-entry.component';
import { ViewBlogEntryComponent } from './components/blog/view-blog-
entry/view-blog-entry.component';
import { PersonalProfileComponent } from './components/user/personal-
profile/personal-profile.component';
```

```
import { ConfigService } from './config.service';
import { UpdateBlogComponent } from './components/blog/update-blog/update-blog.component';
import { MatChipsModule } from '@angular/material/chips';
import { MatAutocompleteModule } from '@angular/material/autocomplete';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { SnackbarComponent } from './components/ui/snackbar/snackbar.component';
import { MatDialogModule } from '@angular/material/dialog';
import { DeleleBlogComponent } from './components/ui/delele-blog/delele-blog.component';
import { StepDirective } from './step.directive';
import { RecipeStepComponent } from './components/ui/recipe-step/recipe-step.component';
import { MatStepperModule } from '@angular/material/stepper';
import { WelcomeMessageComponent } from './components/ui/welcome-message/welcome-message.component';
import { MatExpansionModule } from '@angular/material/expansion';
import { IngredientsComponent } from './components/ui/ingredients/ingredients.component';
import { UserBlogsComponent } from './components/user/user-blogs/user-blogs.component';
import { MatMenuModule } from '@angular/material/menu';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatSidenavModule } from '@angular/material/sidenav';
import { CommonModule } from '@angular/common';
import { DeleteBlogAdminComponent } from './components/ui/delete-blog-admin/delete-blog-admin.component';
import { FullViewImageComponent } from './components/ui/full-view-image/full-view-image.component';
import { MatTabsModule } from '@angular/material/tabs';
import { UserFavoritesComponent } from './components/user/user-favorites/user-favorites.component';
import { MatTooltipModule } from '@angular/material/tooltip';
export function initConfig(config: ConfigService) {
  return () => config.loadConfig();}
@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    RegisterComponent,
    UsersComponent,
```

```
    UserProfileComponent,
    UpdateUserProfileComponent,
    HomeComponent,
    AllBlogEntriesComponent,
    CreateBlogEntryComponent,
    ViewBlogEntryComponent,
    PersonalProfileComponent,
    UpdateBlogComponent,
    SnackbarComponent,
    DeleleBlogComponent,
    StepDirective,
    RecipeStepComponent,
    WelcomeMessageComponent,
    IngredientsComponent,
    UserBlogsComponent,
    DeleteBlogAdminComponent,
    FullViewImageComponent,
    UserFavoritesComponent, ],
  imports: [
    CommonModule,
    // BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatToolbarModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    MatAutocompleteModule,
    MatExpansionModule,
    MatFormFieldModule,
    MatInputModule,
    MatButtonModule,
    MatSelectModule,
    MatTableModule,
    MatPaginatorModule,
    MatCardModule,
    MatChipsModule,
    MatProgressBarModule,
    MatMenuModule,
    MatProgressSpinnerModule,
    MatSnackBarModule,
    MatIconModule,
```

```
    MatSidenavModule,
    MatDialogModule,
    MatTabsModule,
    MatTooltipModule,
    MatStepperModule,
    MarkdownModule.forRoot()
  ],
  providers: [
    ConfigService,
    {
      provide: APP_INITIALIZER,
      useFactory: initConfig,
      multi: true,
      deps: [ConfigService]
    },
    JwtHelperService,
    { provide: JWT_OPTIONS, useValue: JWT_OPTIONS },
    {
      provide: HTTP_INTERCEPTORS,
      useClass: JwtInterceptor,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

— код из сервиса blog.service.ts

```
import { HttpClient, HttpParams } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { BlogDto, BlogEntriesPageable } from 'src/app/model/blog.interface';
@Injectable({
  providedIn: 'root'
})
export class BlogService {
  constructor(private http: HttpClient) { }
  indexAll(page: number, size: number): Observable<BlogEntriesPageable> {
    let params = new HttpParams();

    params = params.append('page', String(page));
```

```
params = params.append('limit', String(size));
  return this.http.get<BlogEntriesPageable>('/api/blogs', {params});
 }
 indexByUser(userId: string): Observable<BlogDto> {
  return this.http.get('/api/blogs/user/' + userId);
 }
 postToModerate(blogDto): Observable<any> {
  return this.http.post<BlogDto>('/api/moderator', blogDto);
 }
 post(blogDto: BlogDto): Observable<BlogDto> {
  console.log('post');
  console.log(blogDto);
  return this.http.post<BlogDto>('/api/blogs', blogDto);
 }
 uploadHeaderImage(formData: FormData): Observable<any> {
  return this.http.post<FormData>('/api/blogs/image/upload', formData, {
    reportProgress: true,
    observe: 'events'
  });
 }
 uploadStepImage(formData: FormData): Observable<any> {
  return this.http.post<FormData>('/api/blogs/image/step-image-upload', formData, {
    reportProgress: true,
    observe: 'events'
  });
 }
 findOneBlog(id: string): Observable<BlogDto> {
  return this.http.get<BlogDto>('/api/blogs/' + id);
 }
 updateUserBlog(blog): Observable<BlogDto> {
  console.log(blog);
  console.log(blog._id);
  return this.http.put('/api/blogs/' + blog._id, blog);
 }
 deleteBlogByUser(blogId): Observable<BlogDto> {
  return this.http.delete('/api/blogs/' + blogId);
 }
 toggle(blog, userId): Observable<boolean> {
  if (blog.likeUserId != 0) {
    blog.likeUserId.forEach(function (v) {
       if(v === userId) { return true; }
     })}
```

```
// if (blog.likeUserid === userId) {
    //   console.log('Айди есть');
    // } else {
    //   console.log('айди нет');
    // }
    console.log(blog);
    console.log(userId);
    return
  }
}
```

— код из компонента all-blog-entries.component.ts

```
import { Component, EventEmitter, Input, OnInit, Output, ViewChild } from
'@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { PageEvent } from '@angular/material/paginator';
import { Router } from '@angular/router';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';
import { BlogDto, BlogEntriesPageable } from 'src/app/model/blog.interface';
import { User } from 'src/app/model/user.interface';
import { AuthenticationService } from 'src/app/services/authentification-
service/authentication.service';
import { BlogService } from 'src/app/services/blog-service/blog.service';
import { FunctionService } from 'src/app/services/function-service/function.service';
import { DeleleBlogComponent } from '../../ui/delele-blog/delele-blog.component';
import { IngredientsComponent } from '../../ui/ingredients/ingredients.component';
@Component({
  selector: 'app-all-blog-entries',
  templateUrl: './all-blog-entries.component.html',
  styleUrls: ['./all-blog-entries.component.scss']
})
export class AllBlogEntriesComponent implements OnInit {
  @Input() blogEntries: BlogEntriesPageable;
  @Output() paginate: EventEmitter<PageEvent> = new EventEmitter<PageEvent>();
  dataSource$: Observable<BlogEntriesPageable> = this.blogService.indexAll(1, 10);
  categorySource$: Observable<any> = this.functionService.getAllCategory();
  // userId$ = this.authService.getUserId();
  // isLiked = this.likeBlog(this.dataSource$.pipe)
  pageEvent: PageEvent;
  isJoined: boolean = false;
```

```
panelOpenState = false;
  constructor(private router: Router,
    private blogService: BlogService,
    private authService: AuthenticationService,
    private functionService: FunctionService,
    public dialog: MatDialog
    ) { }
  ngOnInit(): void {
   const getUserId = this.authService.getUserId();
    if (getUserId !=null) {
      this.isJoined = true;
      this.userId = getUserId
    }
  }
  isLiked = false;
  userId: string = '0';
  value = '';
  allCategories = []
  openDialog() {
    console.log('Modal here');
    const dialogRef = this.dialog.open(IngredientsComponent, {
      width: '100%', height: '70%'
      // data: {name: this.name, animal: this.animal}
    });
    dialogRef.afterClosed().subscribe(result => {
      console.log('The dialog was closed');
      // this.animal = result;
    });
  }
  onPaginateChange(event: PageEvent) {
    console.log(event);
    // let page = event.pageIndex;
    // let limit = event.pageSize;
    // page = page +1;
    event.pageIndex = event.pageIndex + 1;
    // event.previousPageIndex = event.previousPageIndex +1;
    this.paginate.emit(event);
    // console.log(event);
    // this.dataSource = this.blogService.indexAll(page, limit);
    // this.paginate.emit(event)
  }
```

```
navigate(id) {
    this.router.navigateByUrl('blogs/' + id);
  }
  navigateToUpdate(id) {
    this.router.navigate(['update-blog/', id]);
  }
  // likeLog() {
  //   this.isLiked = true
  // }
  likeBlog(blog: BlogDto) {
    const userId = this.authService.getUserId();
    console.log('likeblog');
    this.blogService.toggle(blog, userId)
    // this.blogService.toggle(blog).subscribe(x => { } );
  }
  editBlog(blog) {
    console.log(blog);
  }
  deleteBlog(blog) {
    this.blogService.deleteBlogByUser(blog).subscribe();
  }
}
```

— код из компонента all-blog-entries.component.html

```
<app-welcome-message *ngIf="isJoined === false"></app-welcome-message>
<div class="row row-cols-5 search d-flex justify-content-center align-items-center">
  <div class="col what-to-cook">
    <p>
      Что хотите приготовить?
    </p>
  </div>
  <div class="col">
    <mat-form-field class="example-form-field">
      <mat-label>Название блюда</mat-label>
      <input matInput type="text" [(ngModel)]="value">
      <button mat-button *ngIf="value" matSuffix mat-icon-button (click)="value="">
        <mat-icon>close</mat-icon>
      </button>
    </mat-form-field>
    <!-- <input type="text" placeholder="Любое блюдо, ингридиент или
категория"> -->
```

```
</div>
  <div class="col">
    <mat-form-field appearance="fill">
      <mat-label>Выберите категорию</mat-label>
      <mat-select>
        <mat-option>Любая категория</mat-option>
        <mat-option *ngFor="let category of (categorySource$ | async)"
[value]="category.catTitle">{{category.catTitle}}
        </mat-option>
      </mat-select>
    </mat-form-field>
    <!-- <mat-form-field>
          <mat-select placeholder="Select Level of Education" name="education_level"
(selectionChange)="educationLevelChangeAction(education_level)"
[(ngModel)]="education_level" >
           <mat-option *ngFor="let education of educationList" [value]="education"
>{{ education.educationLevelName }}</mat-option>
          </mat-select>
        </mat-form-field> -->
  </div>
  <div class="col expansion-panel">
    <button mat-stroked-button (click)="openDialog()" class="ingredients-
button">Ингредиенты</button>
  </div>
  <div class="col">
    <button mat-raised-button class="find-button">Подобрать рецепты</button>
  </div>
</div>
<div class="container">
  <h2>Новые рецепты от наших пользователей</h2>
  <div *ngIf="blogEntries?.docs">
    <div class="row row-cols-4">
      <div class="col" *ngFor="let blog of blogEntries?.docs; let i = index">
        <mat-card class="card">
          <div (click)="navigate(blog._id)">
            <img class="blog-image" *ngIf="blog.headerImage; else
placeholderBlogImage"
              src="http://localhost:3000/api/blogs/image/{{blog.headerImage}}">
            <ng-template #placeholderBlogImage>
              <img src="../../../assets/placeholders/placeholder-blog-entry.jpg"
class="blog-image">
            </ng-template>
```

```
<div class="col">
            <small>{{blog.timeForCooking}} минут</small>
            <h2 class="crop-text" style="margin: 0;">
              {{blog.title}}
            </h2>
            <p class="crop-text-2">{{blog.description}}</p>
            <ng-template #placeholderImage>
              <img src="../../../assets/placeholders/placeholder-blog-entry.jpg"
class="user-image">
            </ng-template>
          </div>
        </div>
        <div class="author row">
          <div class="col-4" >
            <img *ngIf="blog.authorId.profileImage; else placeholderImage"
src="http://localhost:3000/api/user/profile-image/{{blog.authorId.profileImage}}"
class="user-image">
          </div>
          <div class="col" >
            <mat-card-subtitle style="margin-top:5px; margin-bottom:
5px;">{{blog.authorId.name}}</mat-card-subtitle>
            <mat-card-subtitle>{{blog.createdAt | date}}</mat-card-subtitle>
          </div>
        </div>
        <div class="bookmark-icon">
          <mat-icon color="accent" matTooltip="Добавить рецепт в избранное"
style="cursor: pointer;" type="button" (click)="addPostToFavorites(post)"
class="bookmark-icon" >bookmark</mat-icon>
        </div>
      </mat-card>
    </div>
  </div>
  <mat-paginator *ngIf="blogEntries?.docs" [length]="blogEntries?.totalDocs"
[pageSize]="blogEntries?.limit"
  [pageSizeOptions]="[5, 10, 50, 100]" (page)="pageEvent = $event;
onPaginateChange($event)" showFirstLastButtons>
 </mat-paginator>
 </div>
</div>
```

Приложение Н

Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание

					Дипломный проект	Лист
Изм.	Лист	№ докум.	Подпись	Дата		38

ОТЗЫВ

на дипломный проект студента
СӘТБАЕВ УНИВЕРСИТЕТІ
по специальности 5В070400 – Вычислительная техника и программное обеспечение
Спириковой Диане Зафаровне
на тему « «Cooking book» - электронная кулинарная книга.»

Цель данного проекта является дать возможность пользователям создавать, изменять, удалять и хранить собственные посты с рецептами в базе данных, а также регистрировать, распределять роли и сохранять данные пользователей в отдельную таблицу в базе данных. Данные, которые будут сохранены в базе данных постов: название поста, описание поста, прикрепленная фотография или фотографии, автор поста, время создания. В базе данных пользователя сохраняются эти данные: имя, никнейм, электронная почта, зашифрованный пароль, роль пользователя (админ, модератор, юзер), фотография пользователя (аватарка), связь с постами, которые делал пользователь. Все элементы и данные должны отслеживать изменения в базе данных и обновляться автоматически.

В техническом плане достоинство данного сайта состоит в быстроте работы, в визуальном плане это удобство дизайна и интерфейса пользователя, так называемом UxUi. Проект был создан с помощью фреймворков NestJs на языке TypeScript, Angular и базы данной MongoDb, которая является noSql системой. Все вышеперечисленные цели и требования были достигнуты.

В результате работы была создана информационная система, в процессе разработки приложение обрело собственный стиль, дополнилось функционалом. Во время разработки были открыты новые возможности фреймворков NestJs и Angular, приняты во внимание многие современные методологии и концепции программирования. Произошло знакомство с облачной нереляционной базой данных MongoDb, которая значительно улучшила качество разработки и скорость работы самого приложения. Замечании к дипломному проекту нет.

Пояснительная записка выполнена на высоком уровне, соответствующим требованиям, предъявляемым к данным видам работ. Считаю, что Спирикова Диана достоина присвоения квалификации бакалавра по направлению информационно коммуникационные технологии по специальности 5В070400 – "Вычислительная техника и программное обеспечение"

Руководитель

PhD, Алибиева Ж.М. _____ «06» июнь 2021 года

StrikePlagiarism.com

SATBAYEV UNIVERSITY

Дата отчета 5/27/2021
Дата редактирования ---

Отчет не был оценен.

Метаданные

Название
Диплом - Спирикова Диана.docx

Автор
Спирикова Диана

Научный руководитель
Жибек Алибиева

Подразделение
ИКиИТ

Список возможных попыток манипуляций с текстом

В этом разделе вы найдете информацию, касающуюся манипуляций в тексте, с целью изменить результаты проверки. Для того, кто оценивает работу на бумажном носителе или в электронном формате, манипуляции могут быть невидимы (может быть также целенаправленное вписывание ошибок). Следует оценить, являются ли изменения преднамеренными или нет.

Замена букв	ß	4
Интервалы	A→	0
Микропробелы	·	0
Белые знаки	ß	0
Парафразы (SmartMarks)	a	0

Объем найденных подобий

Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.

0.26% 0.26% КП1

0.00% 0.00% КП2

0.32% 0.32% КЦ

25
Длина фразы для коэффициента подобия 2

3132
Количество слов

24903
Количество символов

Подобия по списку источников

Просмотрите список и проанализируйте, в особенности, те фрагменты, которые превышают КП №2 (выделенные жирным шрифтом). Используйте ссылку «Обозначить фрагмент» и обратите внимание на то, являются ли выделенные фрагменты повторяющимися короткими фразами, разбросанными в документе (совпадающие сходства), многочисленными короткими фразами расположенные рядом друг с другом (парафразирование) или обширными фрагментами без указания источника ("криптоцитаты").

10 самых длинных фраз

Цвет текста

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**александра шин** Шин Александра **5/8/2019** Satbayev University (ИКиИТ)	8	0.26 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (0.26 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**александра шин** Шин Александра **5/8/2019** Satbayev University (ИКиИТ)	8 (1)	0.26 %

из программы обмена базами данных (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из интернета (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

Список принятых фрагментов (нет принятых фрагментов)

ПОРЯДКОВЫЙ НОМЕР	СОДЕРЖАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Спирикова Диана

Название: Диплом - Спирикова Диана.docx

Координатор: Жибек Алибиева

Коэффициент подобия 1: 0.26

Коэффициент подобия 2: 0.00

Замена букв: 4

Интервалы: 0

Микропробелы: 0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……

27.05.21 ……

Дата

……

Подпись Научного руководителя