|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ "Информатика и управление"

КАФЕДРА "Системы обработки информации"

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

Проектирование и разработка приложения для визуализации данных

Студент ИУК5-81.Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Рахманкулов А.Р

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Вершинин Е.В.

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Антипова О.В.

(Подпись, дата) (И.О.Фамилия)

*2021 г.*

**Аннотация**

Расчетно-пояснительная записка содержит с., рис., табл., источников.

В качестве объекта разработки выступает система визуализации данных, обеспечивающая наглядное представление о событиях.

Целью данной дипломной работы является разработка и реализация frontend и backend приложения для визуализации данных о передвижениях войск Красной Армии во время Великой Отечественной войны 1941-1945 гг.

Выпускная квалификационная работа бакалавра состоит из трех частей: научно-исследовательской, проектно-конструкторской и проектно-технологической.

В научно-исследовательской части были сформулированы задачи проектирования, исследована предметная область и выявлены процессы предметной области, рассмотрены аналоги системы. Также в данном разделе приведено обоснование выбора инструментов и платформы для разработки системы.

В проектно-конструкторской части была спроектирована концептуальная модель предметной области. На основании функциональных требований к системе выделены основные пользовательские сценарии. В результате анализа концептуальной модели предметной области была спроектирована логическая схема базы данных. Также в данном разделе был обоснован выбор архитектуры системы.

В проектно-технологической части описывается реализация функционирующего приложения с помощью выбранных инструментов и платформы разработки. Также в проектно-технологической части были произведены расчет стоимости разработки программного обеспечения и анализ рабочего места на предмет соответствия нормам СанПиН.

Содержание

[Введение 4](#_Toc74034937)

[1. Техническое задание 5](#_Toc74034938)

[2. Научно-Исследовательская часть 12](#_Toc74034939)

[2.1 Постановка задачи проектирования 12](#_Toc74034940)

[2.2 Описание предметной области 12](#_Toc74034941)

[2.3 Анализ аналогов 14](#_Toc74034942)

[2.4 Обоснование выбора инструментов и платформы для разработки 18](#_Toc74034943)

[3. Проектно - конструкторская часть 28](#_Toc74034944)

[3.1 Концептуальная модель предметной области 28](#_Toc74034945)

[3.2 Пользовательские сценарии 29](#_Toc74034946)

[3.3 Логическая схема базы данных 33](#_Toc74034947)

[3.4 Архитектура системы 36](#_Toc74034948)

[4. Проектно-технологическая часть 43](#_Toc74034949)

[4.1 Реализация функционирующего приложения 43](#_Toc74034950)

[4.1.1 Backend приложения 43](#_Toc74034951)

[4.1.2 Frontend приложения 48](#_Toc74034952)

[Заключение 59](#_Toc74034953)

[Список использованных источников 60](#_Toc74034954)

[Приложение А 62](#_Toc74034955)

Введение

Расчетно-пояснительная записка содержит с., рис., табл., источников.

В качестве объекта разработки выступает система визуализации данных, обеспечивающая наглядное представление о событиях.

Целью данной дипломной работы является разработка и реализация frontend и backend приложения для визуализации данных о передвижениях войск Красной Армии во время Великой Отечественной войны 1941-1945 гг.

Выпускная квалификационная работа бакалавра состоит из трех частей: научно-исследовательской, проектно-конструкторской и проектно-технологической.

В научно-исследовательской части были сформулированы задачи проектирования, исследована предметная область и выявлены процессы предметной области, рассмотрены аналоги системы. Также в данном разделе приведено обоснование выбора инструментов и платформы для разработки системы.

В проектно-конструкторской части была спроектирована концептуальная модель предметной области. На основании функциональных требований к системе выделены основные пользовательские сценарии. В результате анализа концептуальной модели предметной области была спроектирована логическая схема базы данных. Также в данном разделе был обоснован выбор архитектуры системы.

В проектно-технологической части описывается реализация функционирующего приложения с помощью выбранных инструментов и платформы разработки. Также в проектно-технологической части были произведены расчет стоимости разработки программного обеспечения и анализ рабочего места на предмет соответствия нормам СанПиН.

# Техническое задание

**Наименование программы**

Веб-приложение визуализации на онлайн-карте данных о передвижениях войск Красной Армии во время Великой Отечественной войны 1941-1945 гг.

**Область применения**

Область применения программного продукта имеет образовательный характер. Создание данной карты позволит визуализировать данные о местоположении войск Красной Армии в период 1941-1945 гг. на основе отчётных донесений подразделений. В результате, появится наглядное представление о передвижении армейских подразделений с начала боевых действий 22 июня 1941 г.

**Объект, в котором используют программу**

Предполагается использовать программу в сети интернет.

**Основание для разработки**

Документы, на основании которых ведется разработка:

ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению»

ГОСТ 34.602-89 «Информационная технология»

ГОСТ 34.601-90 «Автоматизированные системы. Стадии создания»

ГОСТ 7.32-2001 «Межгосударственный стандарт»

Методические указания по выполнению курсовой работы по дисциплине «Разработка программных систем», автором которых является В.Ю. Кириллов

А также на основании настоящего Технического Задания

Организация, утвердившая этот документ: КФ МГТУ им Н. Э. Баумана.

**Назначение разработки**

Разрабатываемое приложение предназначено для ознакомления с визуализированными данными, основанными на исторических документах.

Главной задачей создания данной карты является отображение исторической действительности на основе данных, взятых из ЦАМО (Центральный архив министерства обороны)(archive.mil.ru) в разрезе состояния и расположения армейских подразделений РККА.

**Требования к программе**

**Требования к функциональным характеристикам**

Приложение должно обеспечивать возможность выполнения следующих функций:

* Отображение местоположения войскового подразделения на выбранную дату;
* Отображение сведений о подразделениях (отображение информации о войсковом подразделении);
* Регистрация и авторизация пользователей
* По уровню доступа возможность:
  + Отображение списка командующих
  + Добавление командующего в базу данных
  + Редактирование командующего в базе данных
  + Удаление командующего из базы данных
  + Отображение списка подразделений
  + Добавление войскового подразделения
  + Редактирование войскового подразделения
  + Удаление войскового подразделения
  + Отображение списка городов/местоположений
  + Добавление городов/местоположения
  + Редактирование городов/местоположения
  + Удаление городов/местоположения
  + Отображение актуализированной информации
  + Добавление актуализированной информации о местоположении войскового подразделения на определенную дату
  + Редактирование актуализированной информации о местоположении войскового подразделения на определенную дату
  + Удаление актуализированной информации о местоположении войскового подразделения на определенную дату

Пользовательский интерфейс должен:

* способствовать быстрому освоению инструментов, формированию у него устойчивых навыков использования системы;
* быть спроектирован таким образом, чтобы пользователь вводил информацию естественным образом, не заботясь о ходе вычислительного процесса;

Интерфейс и справочный механизм системы должны:

* обеспечивать возможность лёгкого исправления ошибок ввода, интерфейс клиента не должен требовать повторного ввода данных;
* обеспечить конкретность и понятность пользователю выдаваемой с сервера информации;
* обеспечить согласование объёма представляемой пользователю информации с возможностями его кратковременной памяти.

**Нефункциональные требования**

Для того, чтобы упростить процесс разработки, а также обеспечить возможность совершенствования системы в будущем, необходимо при разработке системы обеспечить выполнение следующих требований:

* Расширяемость
* Масштабируемость
* Кроссплатформенность
* Адаптивный дизайн

**Требования к надежности**

Сервер должен обеспечивать устойчивое функционирование приложения:

* обеспечение работоспособности при одновременном пользовании приложения не менее 1000 человек
* система должна предотвращать ввод некорректных данных

**Условия эксплуатации**

Требования к квалификации и численности персонала

Минимальное количество персонала, требуемого для работы системы, должно составлять не менее 2 штатных единиц – исторический консультант, программист.  В перечень задач, выполняемых консультантом, должны входить:

* Поиск исторических материалов;
* Обработка исторических материалов;
* Создание данных для наполнения приложения.
* В задачи программиста входят:
* написание front-end приложения;
* написание backend приложения;
* сопровождение приложения.

**Требования к составу и параметрам технических средств**

Запуск разработанного Web-приложения и надёжность его работы без сбоев должно обеспечивать аппаратное обеспечение с технические характеристики не мощнееткомпьютера, на котором проводится разработка:

* процессор AMD A8-7410 APU 2.20 GHz;
* 8,00 ГБ оперативной памяти, DDR3;
* видеокарта AMD Radeon R5 Graphics .

Приветствуется изменение указанных характеристик в сторону увеличения. Изменение характеристик в сторону уменьшения нежелательно, поскольку это приведет к значительному снижению работоспособности Web-приложения.

**Требования информационной и программной совместимости**

Приложение должно обеспечивать адаптивный дизайн при переходе на различные устройства.

Веб-сервер должен обеспечивать возможности:

* управления содержанием и структурой передаваемых данных.

**Стадии и этапы разработки**

Таблица 1 - календарный план

|  |  |  |
| --- | --- | --- |
| Стадии | Этапы работ | Дата |
| 1. Формирование требований к АС | 1.1. Обследование объекта и обоснование необходимости создания АС | 17.09.2020 |
| 1.2. Формирование требований пользователя к АС | 17.09.2020 |
| 1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания) | 20.09.2020 |
| 2. Разработка концепции АС | 2.1. Изучение объекта | 22.09.2020 |
| 2.2. Проведение необходимых научно-исследовательских работ | 22.09.2020 |
| 2.3. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя | 25.09.2020 |
| 2.4. Оформление отчета о выполненной работе | 25.09.2020 |
| 3. Техническое задание | 3.1. Разработка и утверждение технического задания на создание АС | 29.09.2020 |
| 4. Эскизный проект | 4.1. Разработка предварительных проектных решений по системе и ее частям | 15.10.2020 |
| 4.2. Разработка документации на АС и ее части | 23.01.2021 |

Продолжение таблицы

|  |  |  |
| --- | --- | --- |
| 5. Технический проект | 5.1. Разработка проектных решений по системе и ее частям | 3.11.2020 |
| 5.2. Разработка документации на АС и ее части | 30.01.2021 |
| 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку | 4.02.2021 |
| 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации | 17.02.2021 |
| 6.Рабочая документация | 6.1. Разработка рабочей документации на систему и ее части | 6.02.2021 |
| 6.2. Разработка или адаптация программ | 26.03.2021 |
| 7. Ввод в действие | 7.1. Подготовка объекта автоматизации к вводу АС в действие | 1.04.2021 |
| 7.2. Подготовка персонала | - |
| 7.3. Комплектация АС поставляемая изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями) | 8.04.2021 |
| 7.6. Проведение предварительных испытаний | 12.04.2021 |
| 7.7. Проведение опытной эксплуатации | 12.04.2021 |

**Порядок контроля и приёмки**

**Методы испытания системы**

Проверка работоспособности сайта на различных веб-браузерах (Mozilla Firefox, Google Chrome, Internet Explorer).

Общие требования к приёмке работ по стадиям

Место проведения приемки работ КФ МГТУ им. Н.Э.Баумана корпус №6.

Прием работы на всех этапах осуществляется непосредственным руководителем курсовой работы.

**Статус приемочной комиссии**

По окончании разработки работоспособное приложение демонстрируется на комиссии.

**Требования к программной документации**

Разработку системы требуется сопровождать техническим заданием, расчетно-пояснительной запиской и графической частью.

Графическая часть должна состоять из:

7 листов формата А1 графического материала.

Презентация.

**Источники разработки**

Задание на выполнение выпускной квалификационной работы бакалавра по направлению «Информатика и вычислительная техника»

# Научно-Исследовательская часть

## Постановка задачи проектирования

Целью данной дипломной работы является разработка и реализация frontend и backend приложения для визуализации данных о передвижениях войск Красной Армии во время Великой Отечественной войны 1941-1945 гг.

Для реализации поставленной цели необходимо решить следующие задачи :

1. Изучить необходимую литературу
2. Изучить аналоги, определить их недостатки и преимущества
3. Определить программное обеспечение и технологии для разработки системы
4. Спроектировать backend приложения
5. Спроектировать frontend приложения
6. Реализовать backend
7. Реализовать frontend
8. Оформить документацию.

## Описание предметной области

Командующий – военный начальник (военачальник) в [вооружённых силах](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%BE%D1%80%D1%83%D0%B6%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D0%BB%D1%8B" \o "Вооружённые силы) [государства](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE), на которого вышестоящим [начальством](https://ru.wikipedia.org/wiki/%D0%92%D0%BB%D0%B0%D1%81%D1%82%D1%8C" \o "Власть) ([глава государства](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B2%D0%B0_%D0%B3%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B0), [Верховный главнокомандующий](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%85%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%BE%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%83%D1%8E%D1%89%D0%B8%D0%B9)) возложено командование [войсками](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B9%D1%81%D0%BA%D0%B0" \o "Войска) (силами), [объединением](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%B2%D0%BE%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D0%BB%D0%BE)" \o "Объединение (военное дело)), [родом войск](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B4_%D0%B2%D0%BE%D0%B9%D1%81%D0%BA) [военных](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%BE%D1%80%D1%83%D0%B6%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D0%BB%D1%8B) сил государства.

Общевойсковая армия – это крупное оперативное войсковое объединение, состоящее из нескольких соединений и отдельных частей различных видов вооружённых сил, родов войск и специальных войск, предназначенное для ведения военных операций.

Армия это крупное воинское формирование оперативного назначения. Армия включает в себя дивизии, полки, батальоны всех родов войск. В составе армии может также находиться один или несколько корпусов. Говорить о структуре и численности армии невозможно, ибо сколько существует или существовало армий, столько и существовало их структур. Обычно штатное звание командующего армией генерал-полковник.

Что касается боевого состава, то в 1941 году армии в основном состояли (армейский комплект) из различных (стрелковых, механизированных) корпусов и частично дивизий (стрелковых, танковых), а также артиллерийских и инженерных частей, а также частей ПВО армейского подчинения [1].

**Описание процессов предметной области**

Попадая на сайт приложения пользователю представляется на карте расположение подразделений западной группировки войск РККА на начало военных действий (22.06.1941).

Пользователю будет предоставлена возможность выбора даты. Также, для определения конкретного подразделения необходимо выводить информацию о нём.

Регистрируясь в системе пользователь получает доступ к функциям управления контентом.

Доступные после регистрации функции включают следующие возможности:

* Добавление командующего в базу данных
* Редактирование командующего в базе данных
* Удаление командующего из базы данных
* Добавление войскового подразделения
* Редактирование войскового подразделения
* Удаление войскового подразделения
* Добавление городов/местоположения на карте
* Редактирование городов/местоположения
* Удаление городов/местоположения на карте
* Добавление актуализированной информации о местоположении

войскового подразделения на определенную дату

* Редактирование актуализированной информации о местоположении войскового подразделения на определенную дату
* Удаление актуализированной информации о местоположении войскового подразделения на определенную дату

## Анализ аналогов

В сети интернет присутствуют материалы о передвижении подразделений во время Великой отечественной войны, но они не интерактивны. Также, в качестве выделения аналогов хотелось бы рассмотреть визуализацию данных с помощью разных инструментов.

В качестве аналогов выделим:

1. Видеоролики по запросу «вторая мировая война на карте» на видеохостинге Youtube.
2. Веб сайт pobediteli.ru

**Видеоролик на Youtube**

В качестве примера работы с визуализацией данных разберем видеоролики по запросу «вторая мировая война на карте» на видеохостинге «Youtube».

Видеохостингом «Youtube» пользуется каждый из нас практически каждый день, поэтому с уверенностью можно сказать, что с интерфейсом взаимодействия мы хорошо знакомы.

Поисковая система выдает по запросу множество видеороликов и потребовалось бы много времени, чтобы проверить их все на критерий информативности. Поэтому разберем первый видеоролик по ссылке: <https://www.youtube.com/watch?v=-iqfn8bOXUI>

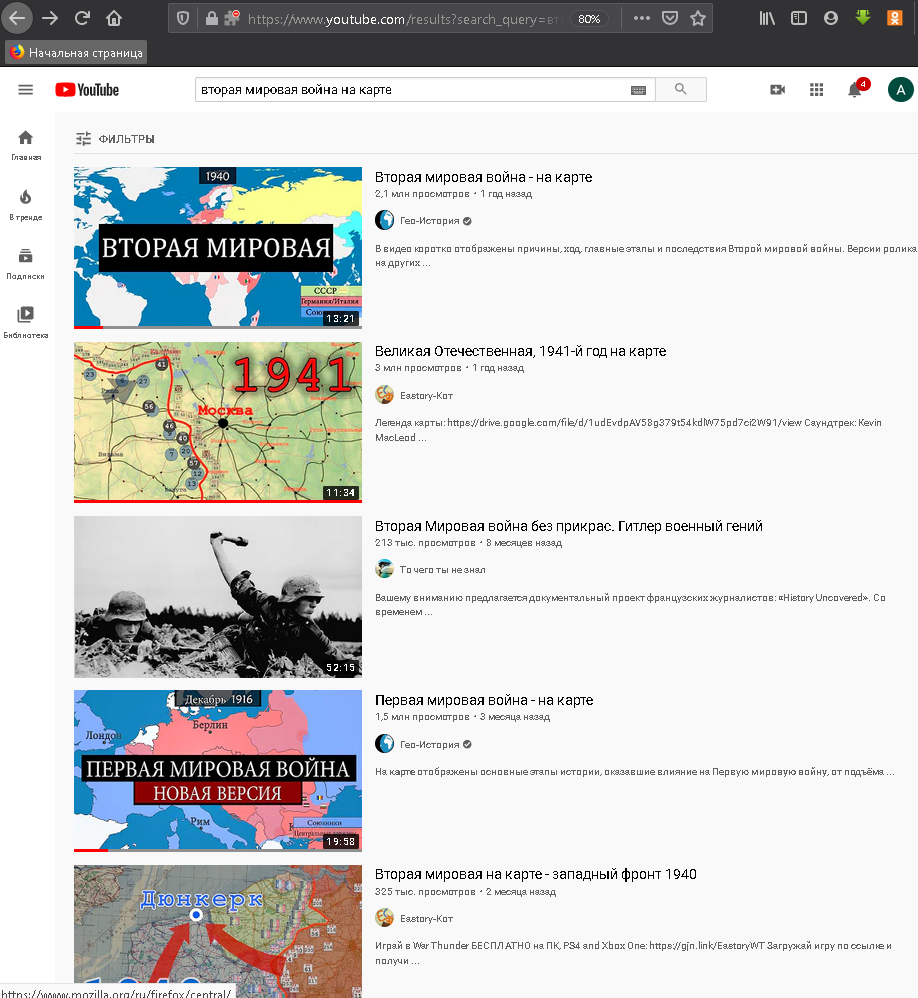


Рисунок – запрос на видеохостинге

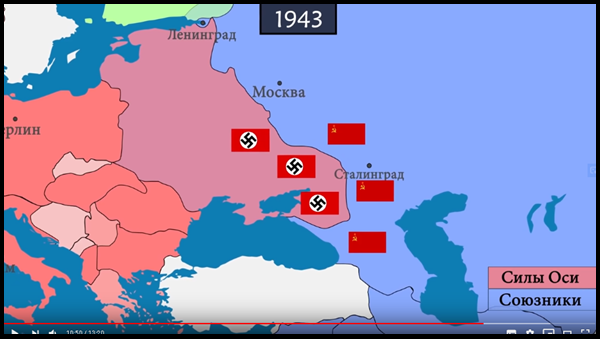


Рисунок – концентрация немецких войск на восточном фронте

Видеоряд изобилует ключевыми событиями. Но детальному разбору не подвергается ни одно из них. Хронометраж видеоролика не позволит уместить такое огромное количество информации, поэтому автору приходилось сокращать повествование.

**Достоинства:**

1. Затрата минимального количества времени на получение информации (13 минут 20 секунд)
2. Понятная и удобная визуализация событий
3. Благодаря аудиосопровождению передается бо́льшая часть информации
4. Доступность информации
5. Присутствует перевод на английском, французском и арабском языках.

**Недостатки:**

1. Поверхностное изложение

Упоминаются ключевые даты, имена и сражения, но обрывочно. Многое упущено.

1. Отсутствие интерактивности

Повествования о событиях ведутся строго по сценарию автора.

1. Существует возможность столкнуться с ненормативной лексикой в комментариях

**Веб-сайт pobediteli.ru**

Проект «ПОБЕДИТЕЛИ» - гражданская инициатива частных лиц и компаний; его цель — выразить благодарность тем, кто победил в Великой Отечественной войне. Сегодня Интернет дает возможность по-новому рассказать, чем была для наших прадедов эта война. Используя визуализированную модель хода войны, разработчики попытались, избегая пафоса и оценок, объединить исторические факты, современные воспоминания и архивные хроники. Авторы проекта надеются, что это не только поможет наглядно представить целостную картину происходившего, но и инициирует диалог между поколениями: воспоминания, включенные в проект — лишь малая часть того, что могут рассказать нам победители.



Рисунок 3 – приложение на сайте pobediteli.ru

**Достоинства:**

1. Исчерпывающая информация об исторических событиях
2. Можно ознакомиться с официальными документами не тратя времени на их поиск в сети интернет.
3. Также в проекте присутствует возможность прослушивания комментариев очевидцев.
4. Удобство использования
5. В приложении присутствует справка о функциональности
6. Возможна навигация по дате и связанными с ней историческими собитиями.
7. Информация черпается из официальных источников
8. Присутствует возможность скачивания программы.
9. возможность скачивания приложения на мобильное устройство
10. Достаточно исчерпывающая информация о самом проекте
11. Есть возможность оставить отзыв
12. Можно принять участие в сопровождении проекта

**Недостатки:**

1. Отсутствует развитие проекта
2. К неявным недостаткам можно отнести тенденциозность

Проект посвящен сбору сведений о ветеранах СССР и стран союзников.

Отсутствуют сведения о немецких подразделениях.

## Обоснование выбора инструментов и платформы для разработки

**С Sharp и DotNet**

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей [2] .

Ещё одним компонентом платформы .NET является технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно относительно быстро разработать веб-приложения, взаимодействующие с базами данных [3].

Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами [4].

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 9.0, которая вышла 10 ноября 2020 года вместе с релизом .NET Core 5.

Объектно-ориентированный подход позволяет строить с помощью C# крупные, но в то же время гибкие, масштабируемые и расширяемые приложения.

C# уже давно поддерживает много полезных функций:

* инкапсуляция,
* наследование,
* полиморфизм,
* перегрузка операторов,
* статическая типизация.

При этом он всё ещё активно развивается, и с каждой новой версией появляется всё больше интересного — например лямбды, динамическое связывание, асинхронные методы и т.д.

У «шарпа» выделяют много преимуществ:

Поддержка подавляющего большинства продуктов Microsoft

Бесплатность ряда инструментов для небольших компаний и некоторых индивидуальных разработчиков — Visual Studio, облако Azure, Windows Server, Parallels Desktop для Mac Pro и др.

Типы данных имеют фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование, так как вы всегда знаете точно, с чем вы имеете дело.

Автоматическая «сборка мусора». Это значит, что нам в большинстве случаев не придётся заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

Низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков.

Но есть у C# и некоторые недостатки:

Приоритетная ориентированность на платформу Windows;

Язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся. Крупной компании покупка лицензионной версии этого языка обойдётся в круглую сумму.

**ASP NET Identity**

ASP.NET Identity представляет встроенную в ASP.NET систему аутентификации и авторизации [5] . Данная система позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями или использовать для входа на сайт учетные записи внешних провайдеров, таких как Facebook.

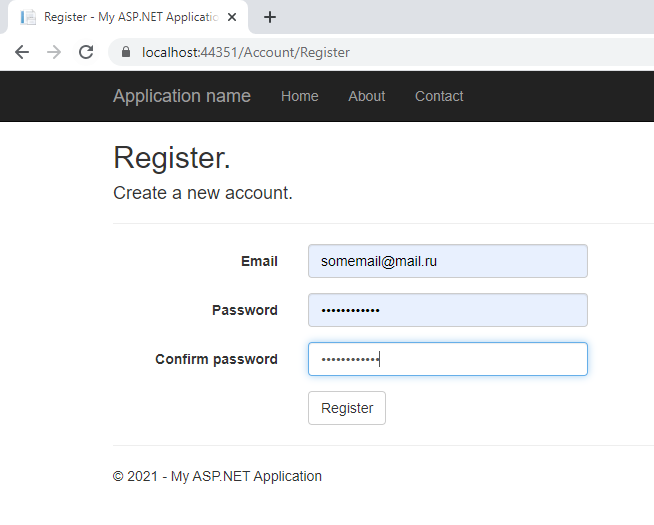


Рисунок – пример регистрации пользователя

Ключевыми объектами в AspNet Identity являются пользователи и роли.

Архитектура ASP NET Identity позволяет взять уже готовый функционал и при необходимости добавить новый, например, добавить для пользователя новое свойство или добавить новую таблицу в базу данных.

Несмотря на то, что по умолчанию нам уже предоставляется готовый функционал, однако в ряде случаев, например, для отправки смс или электронной почты необходима дополнительная настройка.

**Достоинства:**

1. Простота использования
2. Возможность добавления собственного функционала
3. Подробная документация
4. Возможность управления всем процессом

**Недостатки:**

1. Продолжительное время освоения материала
2. Существует вероятность потери времени при изучении смежного материала
3. Вероятность потери времени при отладке приложения, т.к. инструмент не «из коробки»

**JavaScript**

Javascript – динамический скриптовый язык программирования высокого уровня. Он отличается мультипарадигменностью. Речь идет о поддержке функционального, императивного, событийно-ориентированного стилей. Чаще всего язык используется для создания интерактивных веб-страниц и приложений [6] .

Именно в области Frontend задействовано огромное число наработок (компонентов), основанных на Javascript. Наиболее активно используется примерно 25-30 библиотек и фреймворков. Эти готовые шаблоны и решения для стандартных задач (с вариациями) существенно экономят время. Они упрощают процесс web-разработки, ускоряют его, снижая стоимость проектов.

Javascript применяют также для создания мобильных приложений, в серверной (backend) разработке, в десктопных (например, офисных) программах.

**Javascript популярен не случайно, а благодаря своим несомненным достоинствам:**

Незаменимость для веб-разработки. Поддержка скриптов всеми популярными браузерами; полная интеграция с вёрсткой страниц (HTML+CSS) и серверной частью (backend).

Скорость работы и производительность. Javascript позволяет частично обрабатывать веб-страницы на компьютерах пользователя без запросов к серверу. Это экономит время и трафик, снижает нагрузку на сервер.

Лёгкость освоения. Возможно, это дело привычки. Сначала код может показаться сложным, но к его синтаксису, логике быстро привыкаешь. Отображение действий ощутимо добавляет энтузиазма.

**Недостатки Javascript:**

Нет возможности чтения и загрузки файлов. Это ограничение функциональности на стороне клиента. Главная причина – соображения безопасности.

Нестрогая типизация и вольная трактовка. Язык игнорирует явные нестыковки. Имеет место разная интерпретация данных. Нет возможности раннего выявления ошибок. Все недочёты выявляются уже на этапе работы.

Нет поддержки удалённого доступа. Поэтому язык нельзя использовать для сетевых приложений. За это Javascript даже не считают полноценным языком программирования.

Доступность для злоумышленников. В свободный скриптовый язык проще всего встроить фрагмент вредоносного кода, который может навредить пользователю. Надежда только на антивирус и фаервол.

Надо отметить, что сообщество пользователей Javascript активно улучшает язык, устраняя многие недостатки. Узких мест становится всё меньше. Браузеры постоянно совершенствуют работу с JS.

**React**

React.js - [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript)-[библиотека](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript) с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "Открытое программное обеспечение) для разработки [пользовательских интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F" \o "Интерфейс пользователя). React разрабатывается и поддерживается [Facebook](https://ru.wikipedia.org/wiki/Facebook" \o "Facebook), [Instagram](https://ru.wikipedia.org/wiki/Instagram) и сообществом отдельных разработчиков и корпораций. React может использоваться для разработки [одностраничных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Одностраничное приложение) и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость [7].

Material-ui — это набор компонентов React, который реализует Google Material Design (material-ui недавно выпустили v1 библиотеки). Эти компоненты работают изолированно, это означает, что они являются само-поддерживающими и вводят только те стили, которые они должны отображать [8].

**Преимущества React:**

Легок в изучении. React гораздо легче учится ввиду простоты его синтаксиса. Инженеры просто должны вспомнить свои навыки написания HTML и всё на этом. Нет нужды в глубоком изучении TypeScript, как в случае с Angular.

Виртуальная DOM (document object model), которая позволяет упорядочивать документы форматов HTML, XHTML или XML в дерево, которое лучше всего подходит веб-браузерам для анализа различных элементов веб-приложения.

100%-ая JavaScript-библиотека с открытым исходным кодом, которая получает множество ежедневных обновлений и улучшений в соответствии с отзывами разработчиков по всему миру.

**Недостатки React:**

Нехватка официальной документации. Сверхбыстрая разработка React не оставляет места для правильной документации, которая сейчас немного хаотична, так как многие разработчики вносят в неё индивидуальные изменения без какого-либо систематического подхода;

React не имеет чёткой цели. Это означает, что разработчики, иногда, имеют слишком большой выбор;

**Javascript библиотеки для создания интерактивных карт**

Визуализация данных стала неотъемлемой частью жизни практически каждого веб-разработчика.

JavaScript-библиотеки для создания карт можно разделить на два типа. Одни просто позволяют отображать географическое положение одного или нескольких объектов. Для подобной задачи можно использовать карты типа Google Maps или OpenStreetMaps в качестве источника. В принципе, таких решений достаточно, и результат их работы примерно таков:

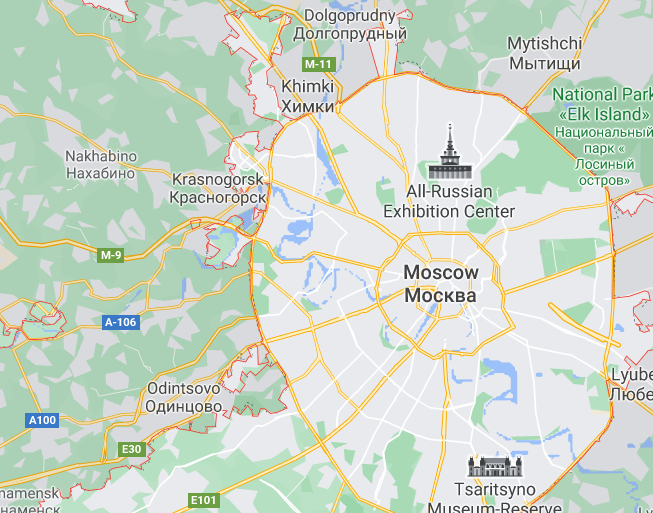


Рисунок – карты google maps

Другой тип – JavaScript-библиотеки, с помощью которых можно делать именно визуализацию данных посредством создания красивых интерактивных карт. Они показывают либо связи между величинами в разных географических регионах, либо перемещение каких-либо объектов относительно их местоположения, и т.п. – всё то, что крайне важно в плане анализа данных и business intelligence.

JavaScript-библиотеки для интерактивных карт-визуализаций

D3.js (от Mike Bostock)

D3.js – мощная в плане возможных результатов библиотека для визуализации данных с открытым исходным кодом. В отличие от других упомянутых JavaScript-библиотек, D3 – это, скорее, фреймворк, который дает максимальную свободу творчества.



Рисунок 6 – пример отображения данных с помощью D3.js

В частности, чтобы сделать такую замечательную интерактивную карту, как на рисунке 6, потребуется изучить достаточно большое количество материала.

Фактически D3 имеет много достоинств, среди которых полная кастомизация поведения, событийная модель и т.д. В целом это идеальное решение для веб-приложений.

Хотя, к сожалению, D3 не предоставляет собственных карт. Это значит – их придется самостоятельно искать в открытых источниках, но вряд ли сейчас это большая проблема.

Документации у D3 нет, однако есть хорошо расписанное API и множество различных туториалов, примеров и прочих полезных материалов буквально по всему Интернету.

Описание API: есть.

Зависимости: нет.

Лицензия: бесплатно.

React-yandex-maps (от Yandex)

react-yandex-maps - это оболочка вокруг API Яндекс.Карт, которая позволяет отображать Яндекс Карты в приложении React с минимальными трудностями.

Эта библиотека представляет собой тонкую оболочку API Яндекс.Карт, многие объекты и функции API Яндекс.Карт поддерживаются компонентами React, но не все из них и не все их функции.

Поддерживаемые функции:

изменение запроса API Яндекс.Карт через компонент провайдера YMaps;

получение доступа к API Яндекс.Карт в компонентах через withYMaps HOC;

гранулярная загрузка необходимых модулей API с помощью API модулей Яндекс.Карт;

контролируемые и неконтролируемые компоненты для каждого поддерживаемого объекта.

Поддерживаемые объекты Yandex.Maps : Map, Panorama.

Поддерживаемые Гео Объекты : GeoObject, Placemark, Polyline, Rectangle, Polygon, Circle, Clusterer, ObjectManager.

Поддерживаемые контролы: Button, FullscreenControl, GeolocationControl, ListBox, ListBoxItem, RouteButton, RouteEditor, RoutePanel, RulerControl, SearchControl, TrafficControl, TypeSelector, ZoomControl

К сожалению, рендеринг узлов React DOM в фабрики шаблонов Yandex.Maps не поддерживается (т. е. нельзя отображать содержимое baloon с помощью React с этой библиотекой из коробки).

Описание API: есть.

Зависимости: API Яндекс карт[[1]](#footnote-1).

Лицензия: бесплатно и платно (от 1600$ в год в зависимости от количества запросов).

Все перечисленные JavaScript-библиотеки для визуализации данных с помощью интерактивных карт – кросс-браузерные. Некоторые из них даже поддерживают старые браузеры, такие как Internet Explorer 6.

Все бесплатные библиотеки довольно-таки неплохо справляются с созданием интерактивных карт. Но они не обеспечивают такую широкую поддержку, как коммерческие.

В результате были выбраны следующие инструменты для разработки:

**Backend:**

Платформа – .Net

Язык программирования – C#

Технология разработки веб-приложения – ASP.NET

API управления пользователями – ASP .NET Identity

**Frontend:**

Язык программирования–Javascript

Библиотека для разработки пользовательского интерфейса–React

Библиотека для создания интерактивных карт – react–yandex-maps

Библиотека стилей – material-ui

# Проектно - конструкторская часть

## Концептуальная модель предметной области

Концептуальная модель отражает семантику предметной области в виде совокупности понятий (сущностей), их характеристик (атрибутов) и связей (ассоциативных отношений между сущностями).

Степень (мощность) связипозволяет определить, сколько экземпляров данной сущности связываются.

В результате анализа предметной области были выделены следующие сущности:

* Командующий
* Звание
* Подразделение
* Тип подразделения
* Удостоверяющий документ
* Местоположение
* Актуальные данные
* Роль
* Пользователь

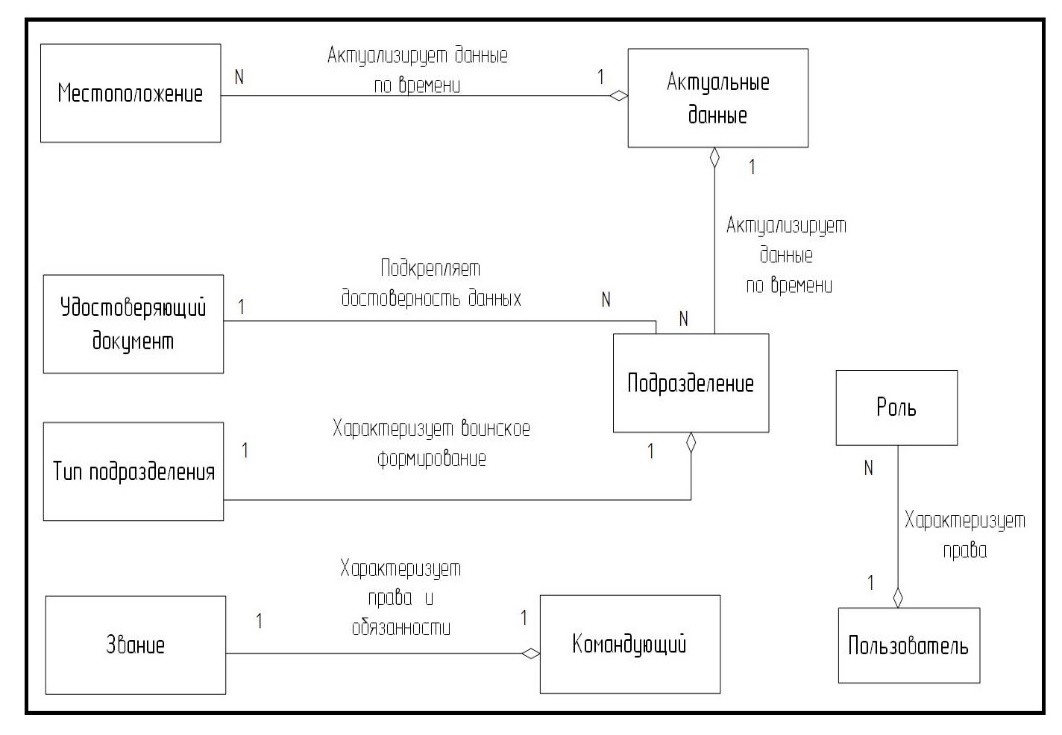


Рисунок – концептуальная модель предметной области

## Пользовательские сценарии

Сценарий использования – это связный рассказ о поведении системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды. Кто-то или что-то может быть пользователем системы, это может быть какая-то информационная система или устройство. И этот кто-то или что-то называется действующим лицом (ДЛ). А сам сценарий использования представляет собой последовательность шагов, которые описывают действия ДЛ и реакцию системы на них[9] .

На основании функциональных требований к системе были выделены основные пользовательские сценарии .

В таблицах 2 – 6 представлены основные пользовательское сценарии поведения системы.

Таблица – сценарий регистрации

|  |  |
| --- | --- |
| Название | Регистрация |
| Краткое описание | Пользователь регистрируется на сайте |
| Участники | Пользователь, frontend, backend |
| Базовый сценарий   1. Пользователь открывает сайт 2. Пользователь просматривает контент и решает помочь развитию проекта 3. Пользователь кликает на кнопку «Регистрация» и перенаправляется на страницу регистрации 4. Пользователь заполняет форму (вводит логин, пароль и подтверждение пароля) и нажимает на кнопку «Зарегистрироваться» 5. Frontend отправляет данные на backend для валидации 6. Приложение перенаправляет пользователя на страницу авторизации | |
| Постусловие | Пользователь зарегистрировался |

Продолжение таблицы 2

|  |  |
| --- | --- |
| Расширения: | |
| 1 | Некорректные данные для регистрации  Форма выдаст сообщение пользователю, что данные введены неверно |
| 2 | Ввод логина, который уже зарегистрирован в системе  Форма выдаст сообщение пользователю,что логин занят |

Таблица – сценарий авторизации

|  |  |
| --- | --- |
| Название | Авторизация |
| Краткое описание | Пользователь авторизуется в системе |
| Участники | Пользователь, frontend, backend |
| Базовый сценарий   1. Пользователь перенаправлен со страницы регистрации на страницу авторизации 2. Пользователь вводит логин и пароль 3. Пользователь решает остаться/не оставаться в системе, чтобы не проходить/пройти процедуру авторизации при следующем посещении сайта и кликает/не кликает на чекбокс «Запомнить меня» 4. Пользователь нажимает на кнопку «Войти» 5. Frontend отправляет данные на backend для валидации 6. Приложение перенаправляет пользователя на главную страницу | |
| Постусловие | Пользователь авторизовался |
| Расширения: | |
| 1 | Некорректные данные для авторизации  Форма выдаст сообщение пользователю, что данные введены некорректно  Пример: пользователь ввёл корректно логин, но введенный пароль и пароль на сервере не совпадают |

Таблица – сценарий создания командующего

|  |  |
| --- | --- |
| Название | Создание командующего |
| Краткое описание | Пользователь создаёт командующего в системе |
| Участники | Пользователь, frontent, backend |
| Базовый сценарий   1. Сценарий доступен только «Зарегистрированному пользователю» 2. Пользователь на сайте выбирает вкладку «Командующие» 3. Пользователь нажимает на кнопку «Добавить» 4. Под кнопкой «Добавить» появляется форма добавления командующего 5. Пользователь вводит Ф.И.О. нового командующего 6. Пользователь выбирает звание командующего 7. Пользователь нажимает на кнопку «Сохранить» 8. Данные с frontend отправляются на сервер для сохранения 9. Пользователю выводится сообщение об успешном добавлении | |
| Постусловие | Пользователь создал командующего |

Таблица – сценарий создания подразделения

|  |  |
| --- | --- |
| Название | Создание подразделения |
| Краткое описание | Пользователь создаёт подразделение |
| Участники | Пользователь, frontent, backend |
| Базовый сценарий   1. Сценарий доступен только «Зарегистрированному пользователю» 2. Пользователь на сайте выбирает вкладку «Подразделения» 3. Пользователь нажимает на кнопку «Добавить» 4. Под кнопкой «Добавить» появляется форма добавления 5. Пользователь вводит название подразделения 6. Пользователь вводит численность подразделения 7. Пользователь вводит состав подразделения 8. Пользователь выбирает командующего ис списка командующих 9. Пользователь выбирает тип подразделения 10. Пользователь нажимает на кнопку «Сохранить» | |

Продолжение таблицы 5

|  |  |
| --- | --- |
| 1. Данные с frontend отправляются на сервер для сохранения   командующего   1. Пользователю выводится сообщение об успешном добавлении | |
| Постусловие | Пользователь создал подразделение в системе |
| Расширения: | |
| 1 | Ввод некорректных данных  Форма выдаст сообщение пользователю, что данные введены некорректно  Пример: пользователь ввел отрицательную численность подразделения |

Таблица – сценарий создания местоположения

|  |  |
| --- | --- |
| Название | Создание местоположения |
| Краткое описание | Пользователь создаёт подразделение |
| Участники | Пользователь, frontent, backend, сторонний сервис |
| Базовый сценарий   1. Сценарий доступен только «Зарегистрированному пользователю» 2. Пользователь на сайте выбирает вкладку «Местоположения» 3. Пользователь нажимает на кнопку «Добавить» 4. Под кнопкой «Добавить» появляется форма добавления 5. Пользователь вводит название местоположения 6. Пользователь кликает на кнопку «Сервис» 7. В открывшемся окне пользователь читает инструкцию и нажимает «Продолжить» 8. Пользователь нажимает «Найти» в верхнем левом углу страницы 9. Пользователь вводит название города и выбирает из списка предложенных 10. Пользователь выбирает масштаб | |

Продолжение таблицы 6

|  |  |
| --- | --- |
| 1. Пользователь находит место и ставит метку 2. В правом нижнем окне пользователь копирует координату X 3. Пользователь переходит на страницу добавления местоположения 4. Пользователь вставляет координату X 5. Пользователь переходит на страницу сервиса 6. Пользователь копирует координату Y 7. Пользователь переходит на страницу добавления местоположения 8. Пользователь вставляет координату Y 9. Пользователь нажимает на кнопку «Сохранить» 10. Данные с frontend отправляются на backend для сохранения 11. Пользователю выводится сообщение об успешном добавлении местоположения | |
| Постусловие | Пользователь создал местоположение в системе |
| Расширения: | |
| 1 | Ввод некорректных данных  Форма выдаст сообщение пользователю, что данные введены некорректно  Пример: пользователь ввел слова в поля координат |

Диаграмма вариантов использования представлена в [Приложении А](#ПриложениеA).

## Логическая схема базы данных

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» связаны между собой внутри системы.

Нотация Чена.

Связь соединяется с ассоциируемыми сущностями линиями. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности [10] .

Таблица – Нотация Чена

|  |  |
| --- | --- |
| **Элемент диаграммы** | **Обозначает** |
|  | независимая сущность |
|  | зависимая сущность |
|  | родительская сущность в иерархической связи |
|  | Связь |
|  | идентифицирующая связь |
|  | Атрибут |
|  | первичный ключ |
|  | внешний ключ (понятие внешнего ключа вводится в реляционной модели данных) |
|  | многозначный атрибут |
|  | получаемый (наследуемый) атрибут в иерархических связях |

В результате анализа концептуальной модели предметной области были сформированы сущности и атрибуты базы данных:

Таблица – сущности и атрибуты базы данных

|  |  |
| --- | --- |
| Название сущности | Атрибуты |
| Пользователь | Идентификатор пользователя (первичный ключ)  Идентификатор роли (внешний ключ)  Логин  Пароль |
| Роль | Название роли  Идентификатор роли (первичный ключ) |
| Звание | Идентификатор звания(первичный ключ)  Название |
| Командир | Идентификатор командира (первичный ключ)  Идентификатор звания ( внешний ключ)  Ф.И.О(составной атрибут) |
| Удостоверяющий документ | Идентификатор удостоверяющего документа(первичный ключ)  Название документа |
| Тип подразделения | Идентификатор типа подразделения(первичный ключ)  Название |
| Подразделение | Идентификатор подразделения(первичный ключ)  Численность  Название  Состав подразделения  Идентификатор командира(внешний ключ)  Идентификатор типа подразделения(внешний ключ) |
| Местоположение | Идентификатор местоположения(первичный ключ)  Название  Координата X  Координата Y |

Продолжение таблицы 8

|  |  |
| --- | --- |
| Актуальные данные | Идентификатор актуальных данных(первичный ключ)  Идентификатор подразделения(внешний ключ)  Идентификатор местоположения(внешний ключ)  Идентификатор удостоверяющего документа(внешний ключ)  Страница документа  Дата |

Логическая схема базы данных представлена в [Приложении А.](#ПриложениеA)

## Архитектура системы

Для того, чтобы обеспечить гибкость разрабатываемой системы, необходимо разделить всю ее функциональность на относительно независимые части. Классической реализацией данного подхода является многослойная архитектура приложения.

Многослойная архитектура обеспечивает группировку связанной функциональности приложения в разных слоях, выстраиваемых вертикально, поверх друг друга. Правильное разделение приложения на слои помогает поддерживать строгое разделение функциональности, что в свою очередь, обеспечивает гибкость, а также удобство и простоту обслуживания.

При строгом разделении на слои, компоненты одного слоя могут взаимодействовать только с компонентами того же слоя или компонентами слоя, расположенного прямо под данным слоем. Более свободное разделение на слои позволяет компонентам слои взаимодействовать с компонентами того же и всех нижележащих слоев.

Частным случаем многослойной архитектуры является MVC (model view controller) архитектура.

MVC - схема разделения данных приложения, [пользовательского интерфейса](https://ru.wikipedia.org/wiki/Пользовательский_интерфейс" \t "Пользовательский интерфейс) и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. Общий вид архитектуры представлен на рисунке 8.



Рисунок – схема паттерна MVC

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения контроллера. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя, а также модели представлений, контроллеры, объекты контекста пользовательских запросов.
* Контроллер (Controller) содержит инструменты, которые отвечают за обработку полученных от уровня представлений данных. На данном уровне находится вся бизнес-логика приложения, а также инструменты взаимодействия с базой данных [4].

На рисунке 8 модель является независимым компонентом - любые изменения контроллера или представления никак не влияют на модель. Контроллер и представление являются относительно независимыми компонентами. Так, из представления можно обращаться к определенному контроллеру, а из контроллера генерировать представления, но при этом нередко их можно изменять независимо друг от друга.

Такое разграничение компонентов приложения позволяет реализовать концепцию разделение ответственности, при которой каждый компонент отвечает за свою строго очерченную сферу. В связи с чем легче построить работу над отдельными компонентами. И благодаря этому приложение легче разрабатывать, поддерживать и тестировать отдельные компоненты.

**Клиент-серверная архитектура**

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций)[11].

Веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер.

Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный пользователю.

Клиент – это браузер, но встречаются и исключения (в тех случаях, когда один веб-сервер выполняет запрос к другому , роль клиента играет веб-сервер). В классической ситуации (когда роль клиента выполняет браузер) для того, чтобы пользователь увидел графический интерфейс приложения в окне браузера, последний должен обработать полученный ответ веб-сервера, в котором будет содержаться информация, реализованная с применением HTML, CSS, JS (самые используемые технологии). Именно эти технологии «дают понять» браузеру, как именно необходимо «отрисовать» все, что он получил в ответе.

Веб-сервер – это сервер, принимающий HTTP-запросы от клиентов и выдающий им HTTP-ответы. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает. Наиболее распространенными видами ПО веб-серверов являются Apache, IIS и NGINX.

База данных - это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категоризировать. Базы данных функционируют под управлением так называемых систем управления базами данных (далее – СУБД). Самыми популярными СУБД являются MySQL, MS SQL Server, PostgreSQL, Oracle (все – клиент-серверные).

Трехзвенная архитектура представляет собой сетевое приложение, которое разделено на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате.

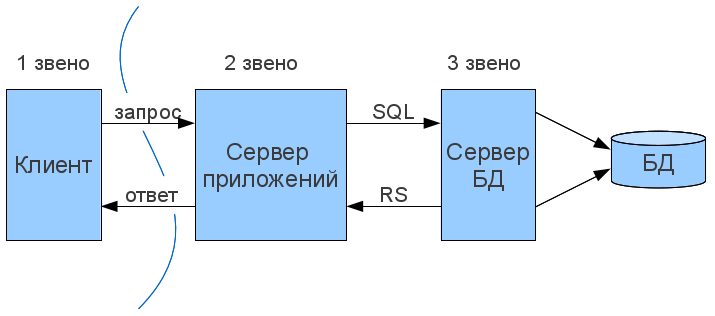


Рисунок – модель трехзвенной архитектуры

Третьим звеном в трехзвенной архитектуре становится сервер приложений, т.е. компоненты распределяются следующим образом:

Представление данных — на стороне клиента.

Прикладной компонент — на выделенном сервере приложений (как вариант, выполняющем функции промежуточного ПО).

Управление ресурсами — на сервере БД, который и представляет запрашиваемые данные.

**RestApi**

API (Application Programming Interface) обеспечивает взаимодействие между двумя системами.

API часто получают данные из пользовательских интерфейсов. При нажатии на кнопку в интерфейсе, запускаются внутренние функции, чтобы передать и получить информацию. Но вместо того, чтобы получать информацию из одной и той же системы, веб-API вызывают удаленные сервисы в сети, чтобы получить их информацию.

Веб-сервисы - API, использующие протокол HTTP .

«Веб-сервис» - это веб-приложение, предоставляющее ресурсы в формате, используемом другими компьютерами. Веб-сервисы - это, в основном, запросы и ответы между клиентами и серверами (компьютер запрашивает ресурс, а веб-сервис отвечает на запрос).

API, использующие протокол HTTP для передачи запросов и ответов, рассматриваются как «веб-сервисы». В случае веб-сервисов клиент, делающий запрос на ресурс, и сервер API, предоставляющий ответ, могут использовать любой язык программирования или платформу. Не имеет значения, какой ЯП или платформа будут использоваться, потому что запрос сообщения и ответ сделаны через общий веб-протокол HTTP.

Веб-протокол является частью прекрасного веб-сервисов: они независимы от языка и поэтому совместимы между различными платформами и системами. REST (Representational State Transfer) использует HTTP в качестве протокола передачи данных для запросов и ответов. При документировании REST API не имеет значения, строят ли инженеры API с помощью С#, Java, Ruby, Python или какого-либо другого языка. Запросы выполняются через HTTP, и ответы возвращаются через HTTP.

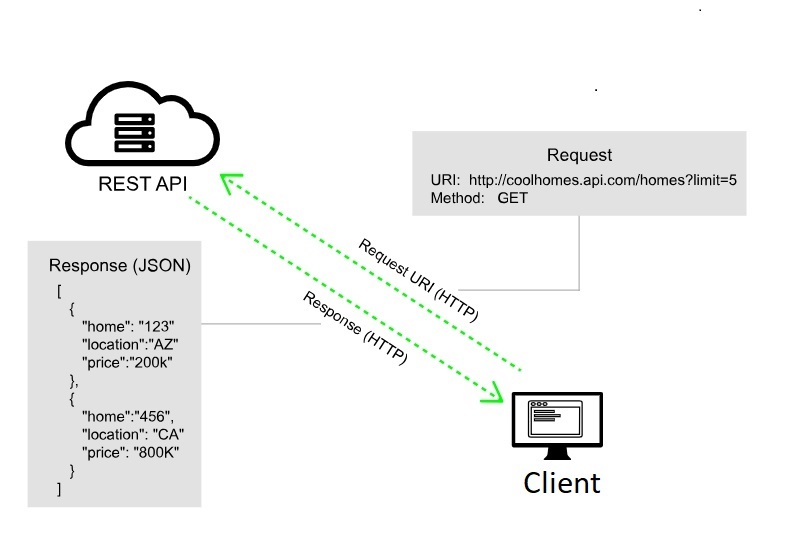


Рисунок – модель REST API

Общие методы включают GET (чтение), POST (создание), PUT (обновление) и DELETE (удаление). Конечная точка обычно включает параметры запроса, которые определяют более подробную информацию о представлении ресурса, который нужно увидеть [12].

REST API не сохраняют свои состояния и могут кэшироваться. Отсутствие состояния означает, что каждый раз, когда вы обращаетесь к ресурсу через конечную точку, API предоставляет один и тот же ответ. Он не запоминает ваш последний запрос и не учитывает его при предоставлении нового ответа. Другими словами, нет ранее запомненных состояний, которые API учитывает при каждом запросе.

Ответы могут кэшироваться для повышения производительности. Если кэш браузера уже содержит информацию, запрашиваемую в запросе, браузер может просто вернуть информацию из кэша вместо того, чтобы снова стучаться на сервер.

Кэширование API REST аналогично кешированию веб-страниц. Браузер использует значение времени последнего изменения в заголовках HTTP, чтобы определить, нужно ли ему снова получать ресурс. Если содержимое не было изменено с момента последнего извлечения, вместо него можно использовать кэшированную копию. Кэширование увеличивает скорость ответа

В основе любого REST API лежит запрос и ответ, передаваемые через Интернет.

**Прототип интерфейса**

На основе функциональных требований был разработан прототип интерфейса приложения.

Область «Header» должна содержать элементы авторизации и регистрации в приложении.

В области «Body» должен располагаться основной контент. Эта область включает в себя компоненты «Date seleсtion» и «Map».

В компоненте «Date selection» должен происходить выбор даты для отображения местоположения подразделения.

В компоненте «Map» должна располагаться карта. Внутри карты должно отображаться местоположение подразделений и информация о конкретном подразделении.

В области «Footer» должны находиться навигационные элементы приложения.

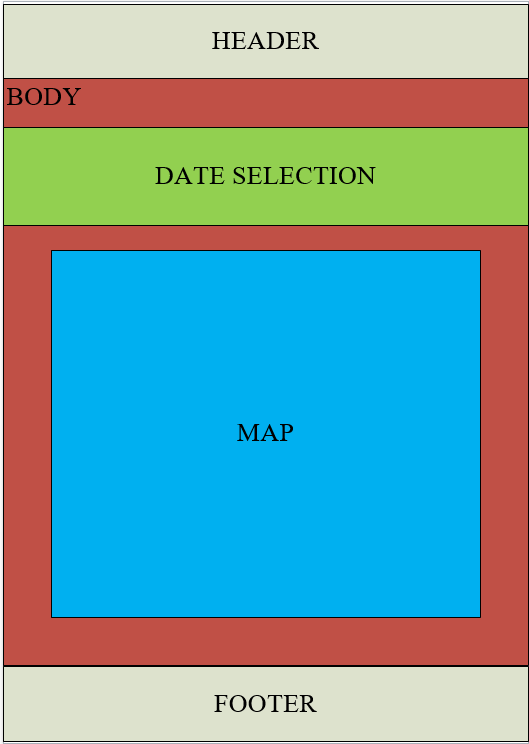


Рисунок – прототип главной страницы приложения

# Проектно-технологическая часть

## Реализация функционирующего приложения

### Backend приложения

Одной из главной точкек входа в микросервис является приложение, которое слушает REST-запросы от клиента по протоколу HTTP. Данное приложение представляет обычное приложение ASP .NET Core. Все основные настройки приложения происходят в классе Startup, представленном на рисунке 12 .

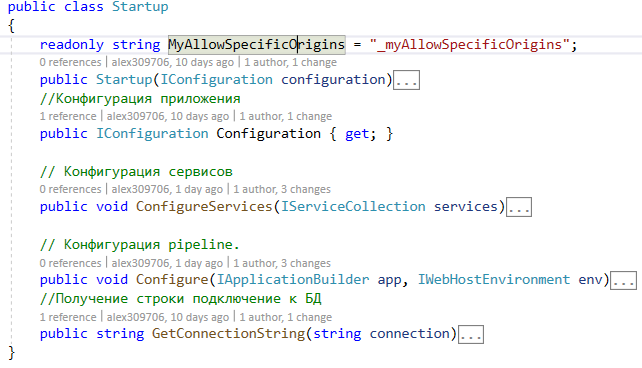


Рисунок 12 – класс Startup

Данное приложение использует в качестве удобного инструмента инъекцию зависимостей (Dependency Injection - DI), которая позволяет в одном месте зарегистрировать все необходимые сервисы, после чего, из любой точки программы мы сможем иметь доступ к экземплярам зарегистрированных случаев, с настройками, которые мы указали. Методом конфигурации сервисов является ConfigureServices(IServiceCollection services), с параметром, который представляет коллекцию всех сервисов. Так, в свою очередь все регистрируемые сервисы представлены на рисунке 13.



Рисунок 13 – конфигурация сервисов

Данный метод добавляет в контейнер сервисы по взаимодействию с базой данных, контроллерами, CORS[[2]](#footnote-2) , swagger[[3]](#footnote-3) .Это нам позволит легко получать доступ к экземпляру сервиса из любой точки программы.

Все запросы, которые будут приходить на backend, будут обрабатываться определенным конвейером, после чего по маршруту запроса будет вызываться определенный метод контроллера. Все контроллеры находящиеся в папке Controllers, представленны на рисунке 14.

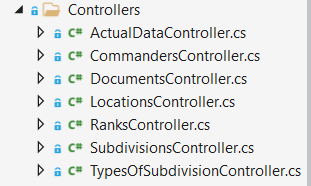


Рисунок 14 – контроллеры

Все что находится перед постфиксом Controller будет участвовать в как часть маршрута запроса. Рассмотрим контроллер, который работает с актуализированными данными (ActualDataController), представленный на рисунке 15.

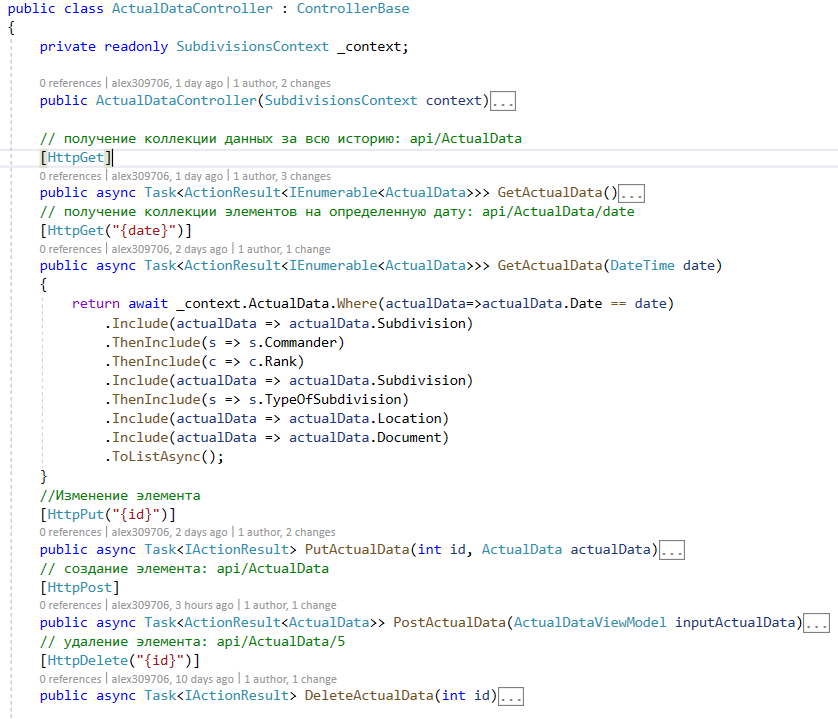


Рисунок 15 – контроллер ActualData

При запросе на адрес [https://localhost:44315/api/ActualData/{date}](https://localhost:44315/api/ActualData/%7bdate%7d) будет вызван метод GetActualData(DateTime date). Этот метод вернет коллекцию элементов актуальных данных (ActualData) на определенный период времени, заданный во входном параметре date. В самом методе GetActualData с помощью методов расширения технологии LINQ[[4]](#footnote-4) формируется запрос на возвращение актуальных данных из записей в базе данных, свойство Date которых совпадает с date из входных параметров. Далее происходит подгрузка данных по подразделению, командиру подразделения, звания командира, типа подразделения, местоположения подразделения и документа, на основании которого происходит запись в базу данных.

Модель объекта актуальных данных содержит информацию о подразделениях и их расположении на определенный момент времени, а также ссылку на документ, на основании которого была осуществлена запись в базу данных. Свойства объекта представлены на рисунке 16.

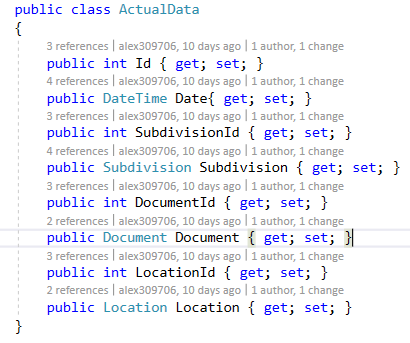


Рисунок 16 – модель объекта ActualData

Также проект содержит контроллеры для работы с другими моделями:

CommandersController: предоставляет методы для работы с командующими. Получение списка командующих; создание, редактирование и удаление командующего.

DocumentsController: предоставляет методы для работы с удостоверяющими документами. Получение списка документов; создание, редактирование и удаление документов.

LocationsController: предоставляет методы для работы с местоположениями. Получение списка местоположений; создание, редактирование и удаление местоположения.

RanksController: предоставляет методы для работы с званиями. Получение списка званий; создание, редактирование и удаление звания.

SubdivisionsController: предоставляет методы для работы с подразделениями. Получение списка подразделений; создание, редактирование и удаление подразделения.

TypesOfSubdivisionController: предоставляет методы для работы с типами подразделения. Получение списка типов подразделений; создание, редактирование и удаление типа подразделения.

Для удобного взаимодействия с API используется Swagger, который на основе информации о типах создаёт страничку по определенному адресу, по которому можно удобно просматривать все методы API, даже производить тестирование рисунок 17.

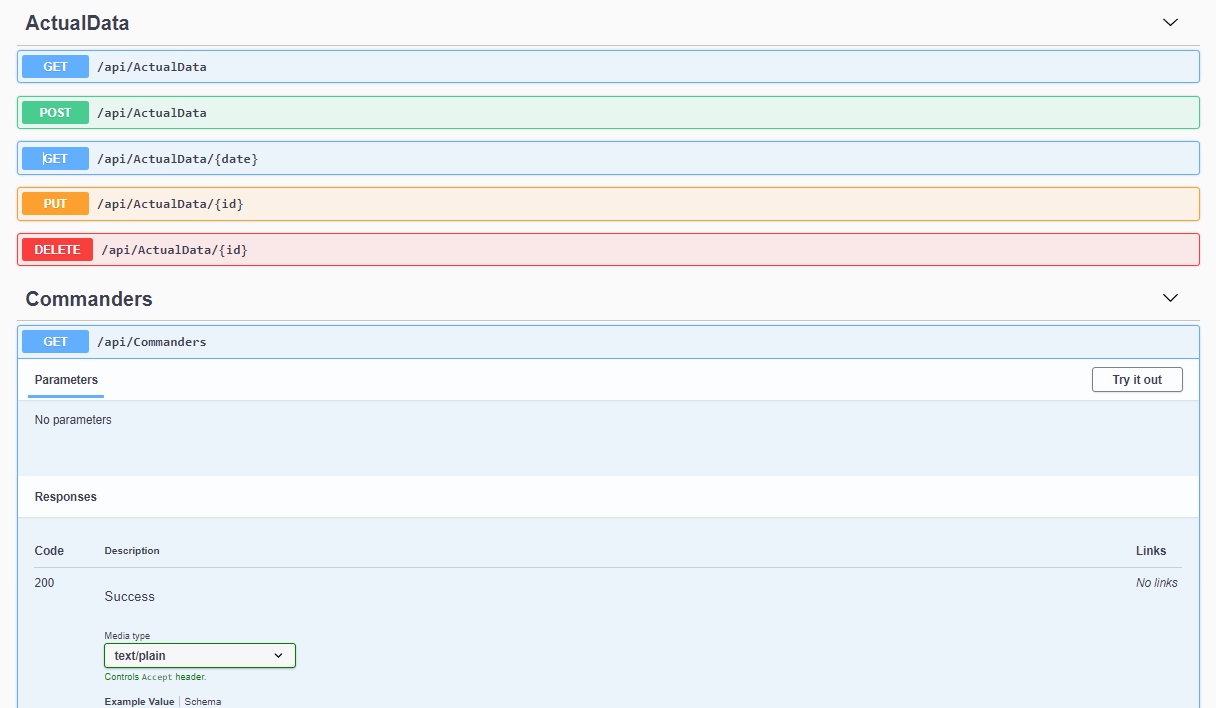


Рисунок 17 – интерфейс swagger’а

Таким образом получаем, что backend реализует работу, направленную для работы зарегистрированным пользователем с актуальными данными, командующими, документами, местоположениями, званиями, подразделениями и типами подразделений.

### Frontend приложения

Точкой входа в приложение React является метод index.js.. В нем указывается, что нужно отрендерить компонент App в DOM[[5]](#footnote-5) объекте с идентификатором root. Метод представлен на рисунке 18.



Рисунок 18 – метод index.js

Объект, в котором необходимо отрендерить содержимое компонента App представляет собой блочный элемент div, который находится в секции body. Именно в этой части размещается содержиние документа, которое видит пользователь.



Рисунок 19 – страница index.html

Функциональный компонент App представлен на рисунке 20.



Рисунок 20 – компонент App

Router определяет набор маршрутов и, когда к приложению, приходит запрос, то Router выполняет сопоставление запроса с маршрутами. И если какой-то маршрут совпадает с URL запроса, то этот маршрут выбирается для обработки запроса.

И также для выбора маршрута определен объект Switch. Он позволяет выбрать первый попавшийся маршрут и его использовать для обработки. Без этого объекта Router может использовать для обработки одного запроса теоретически несколько маршрутов, если они соответствуют строке запроса.

Каждый маршрут представляет объект Route. Он имеет ряд атрибутов. В частности, здесь для маршрута устанавливаются два атрибута:

path - шаблон адреса, с которым будет сопоставляться запрошенный адрес URL

component - тот компонент, который отвечает за обработку запроса по этому маршруту.

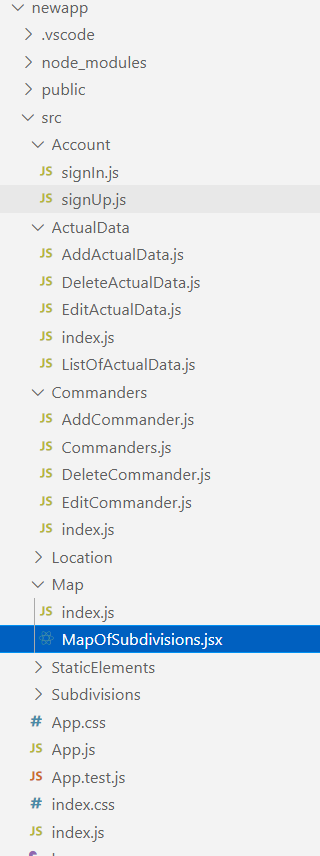


Рисунок – древо элементов проекта React

Для регистрации пользователя в системе разработан компонент SignUp.

Пользователю необходимо придумать логин и пароль и ввести их в соответствующие поля ввода. Процедура повторного ввода пароля позволит системе убедиться,что пользователь запомнил пароль.

Далее пользователю необходимо нажать на кнопку «Зарегистрироваться».

В случае успешной регистрации пользователь будет перенаправлен на главную страницу сайта приложения.

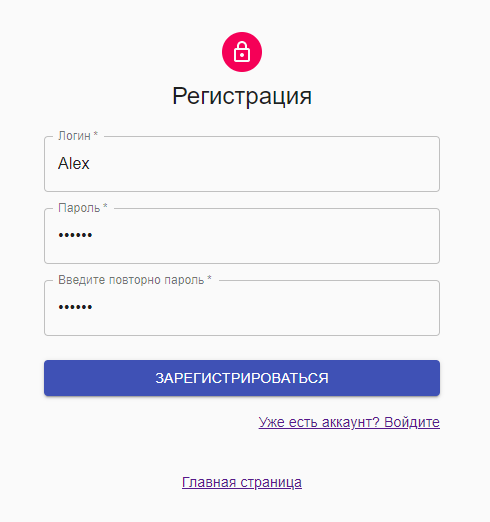


Рисунок - форма регистации

Присутствует возможность перехода на форму авторизации, если пользователь уже зарегистрирован в системе. Эту задачу выполняет компонент signIn.

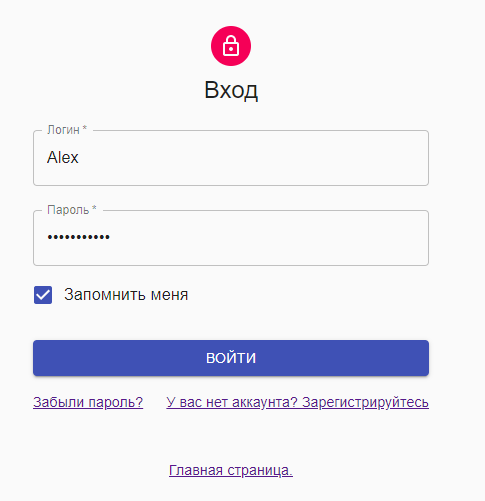


Рисунок 23 - форма авторизации

Для выхода из аккаунта достаточно кликнуть на поле «Выйти» из списка предложенных опций.

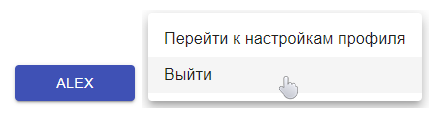


Рисунок 24 – кнопка профиля

Для навигации по приложению был сформирован элемент, представленный на рисунке 25.



Рисунок 25 – навигационный элемент в области "Footer"

Зарегистрированному пользователю доступны функции добавления, изменения и удаления записей из базы данных системы согласно функциональным требованиям к системе.

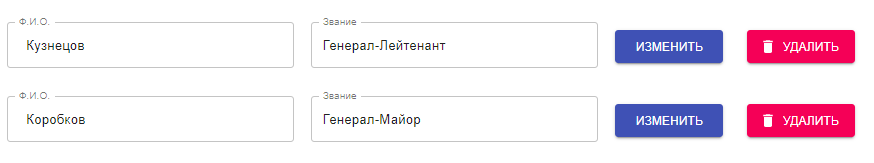


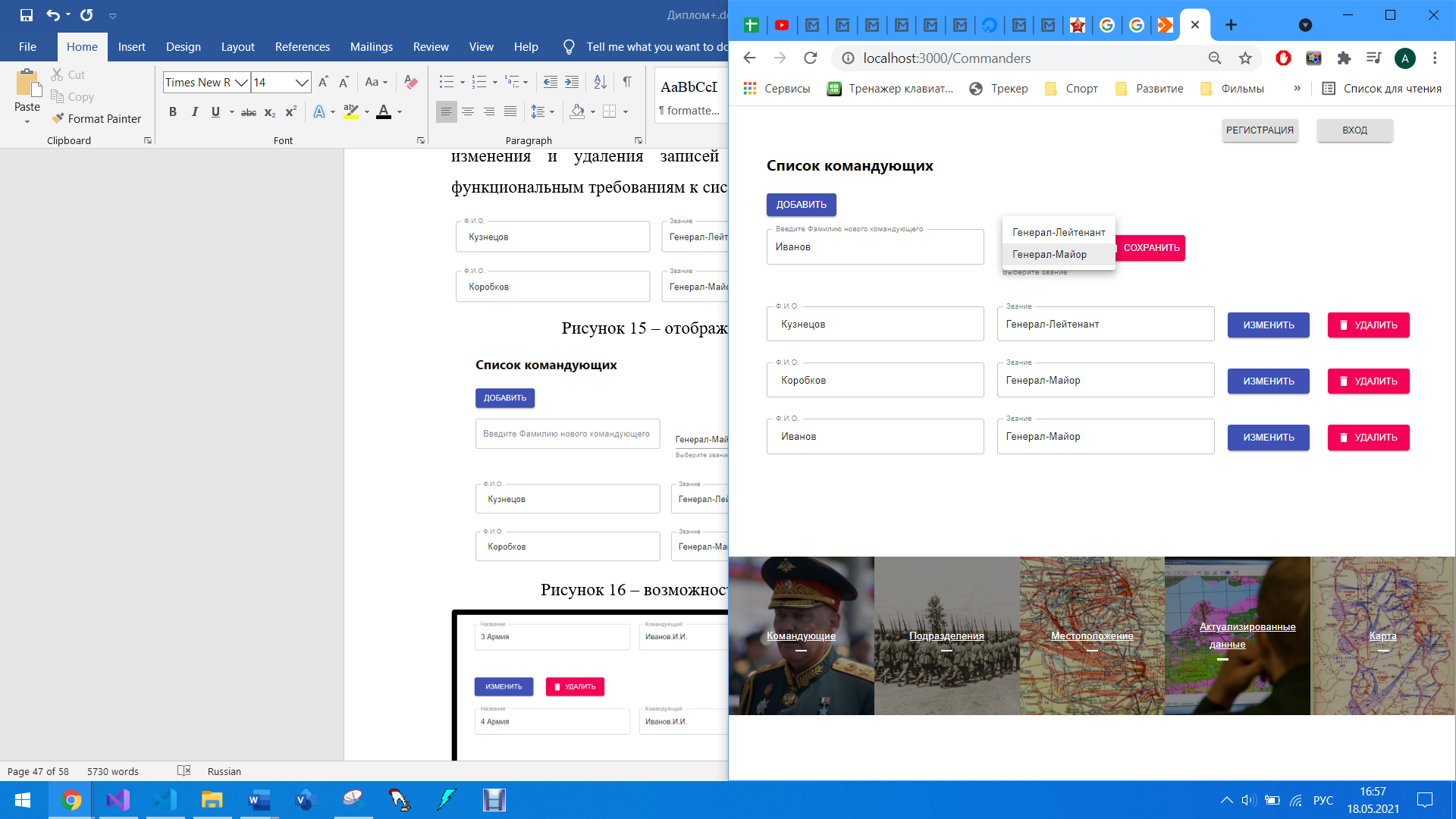
Рисунок 26 – отображение списка командующих

Рисунок 27 – возможность добавления командующего



Рисунок 28 – вывод списка подразделений

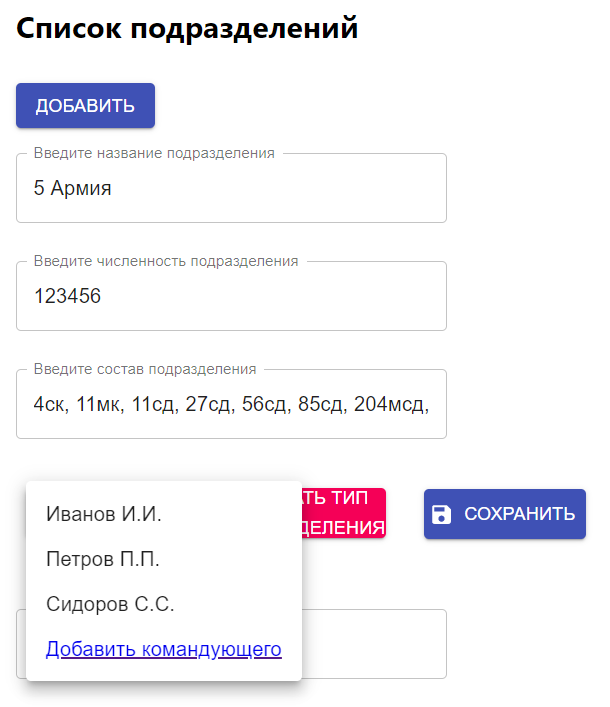


Рисунок 29 – добавление подразделения

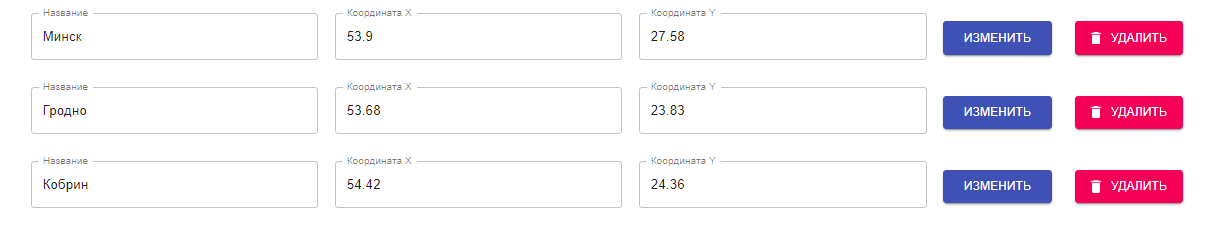


Рисунок 30 – отображение списка местоположений

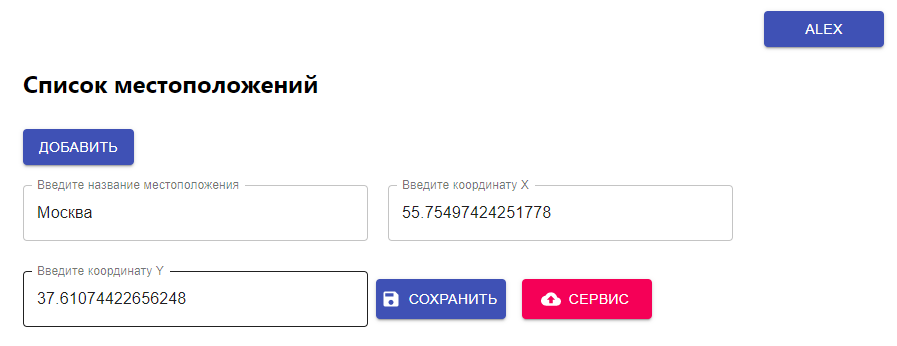


Рисунок 31 – добавление местоположения

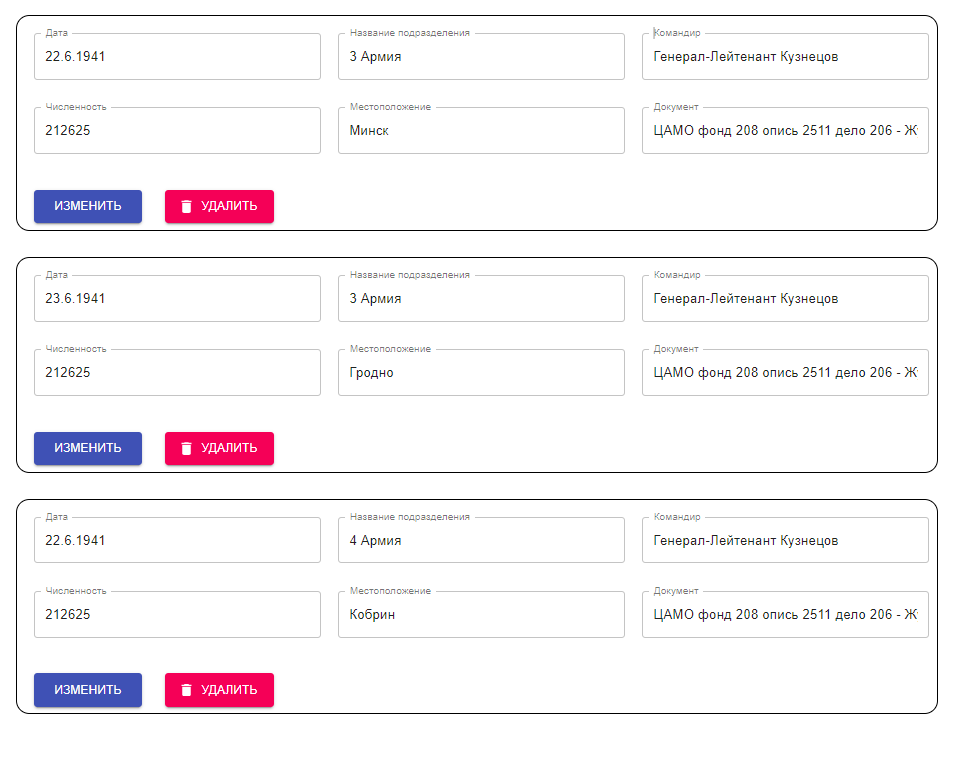


Рисунок 32 - вывод актуальных данных

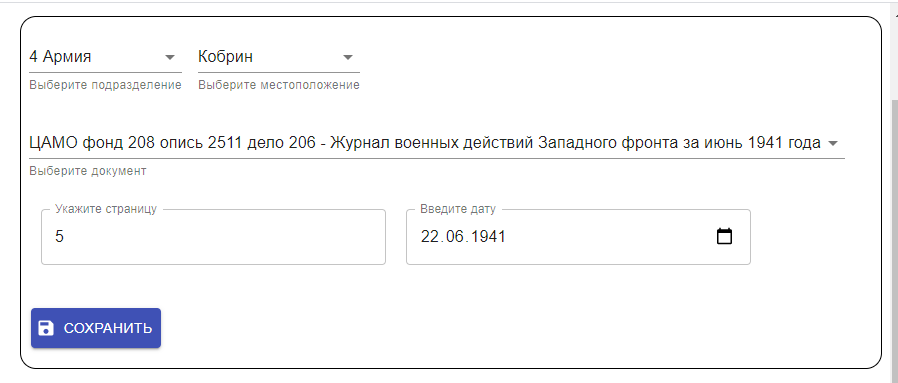


Рисунок 33 – добавление актуальных данных

Для отображения подразделений на карте был определен компонент MapOfSubdivisions.

Для хранения значения адреса к Api была введена константа UrlToApi.

Хуки — нововведение в React 16.8, которое позволяет использовать состояние и другие возможности React без написания классовых компонентов [7].

Хук useState предоставляет функциональным компонентам доступ к состоянию React.

Для хранения значений состояния были введены переменные состояния:

DateTime с возможностью обновления значения при помощи функции setDateTime. Переменная инициализирована значением «1941-06-22»

ActualData с возможностью обновления значения при помощи функции setActualData. Переменная инициализирована значением пустым массивом.

Url с возможностью обновления значения при помощи функции setUrl. Переменная инициализирована значением 'https://localhost:44315/api/ActualData/1941-06-22'.



Рисунок 34 – компонент MapOfSubdivisions

Для заполнения данными объект ActualData используется функция FetchActualData. В этом методе с помощью axios[[6]](#footnote-6) выполняется GET запрос на backend. Backend отправляет обратно коллекцию элементов ActualData, если на выбранную дату существуют актуальные данные. В этом можно убедиться открыв инструменты разработчика в браузере и перейдя на вкладку «Network».

Пример выполнения данной операции представлен на рисунке 35.

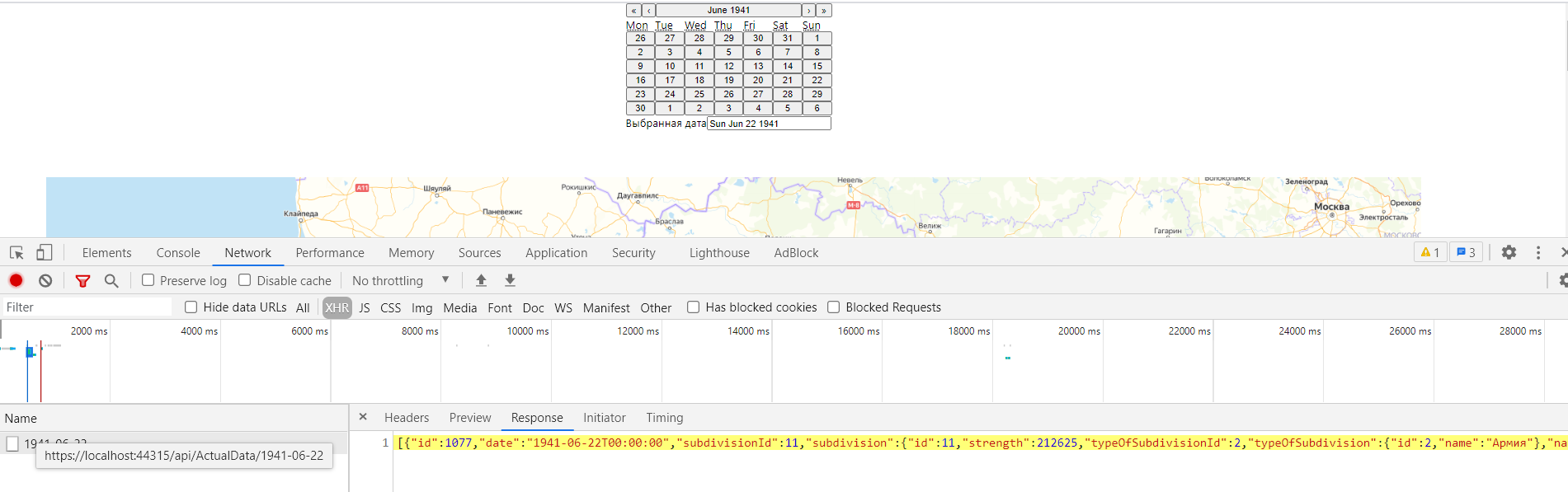


Рисунок 35 – проверка ответа

Ответ сервера записывается в переменную состояния ActualData.

Хук useEffect срабатывает сразу же при инициализации переменной состояния Url. Следовательно объект состояния ActualData заполняется данными.

Хук эффекта даёт возможность выполнять побочные эффекты в функциональном компоненте.

Если данные в базе данных присутствуют, возвращается компонент div с вложенными в него компонентами CustomCalendar и YMaps.

CustomCalendar представляет собой стилизованный компонент Calendar из библиотеки react-calendar с определенным начальным значением и переданным в него событием обработчика нажатия смены даты из родительскиго компонента.

Обработчик смены даты представляет собой событие, которое принимает в качестве входного параметра дату в полной форме (день недели, месяц, день месяца, год, время, часовой пояс).

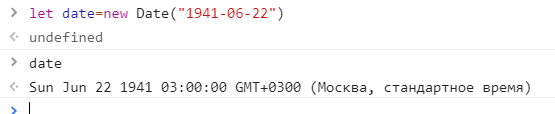


Рисунок 36 – пример полного описания даты

Далее происходит извлечение данных с помощью методов getFullYear(),

getMonth(), getDate().

Из извлеченных данных формируется строка, представляющая дату.

С помощью метода setDatetime устанавливается значение переменной состояния Date.

С помощью метода setUrl устанавливается значение переменной состояния Url.

Объект YMaps представляет собой обёртку для работы с картами.

Объект Map представляет собой карту. а

С помощью объекта ObjectManager можно создавать объекты на карте. В данном случае в параметр features, представляющий собой тип FeatureCollection, передается массив объектов типа Feature со свойствами, представляющими свойства объекта актуальных данных из базы данных.

Схема взаимодействия пользователя с системой представлена в [Приложении А](#ПриложениеA).

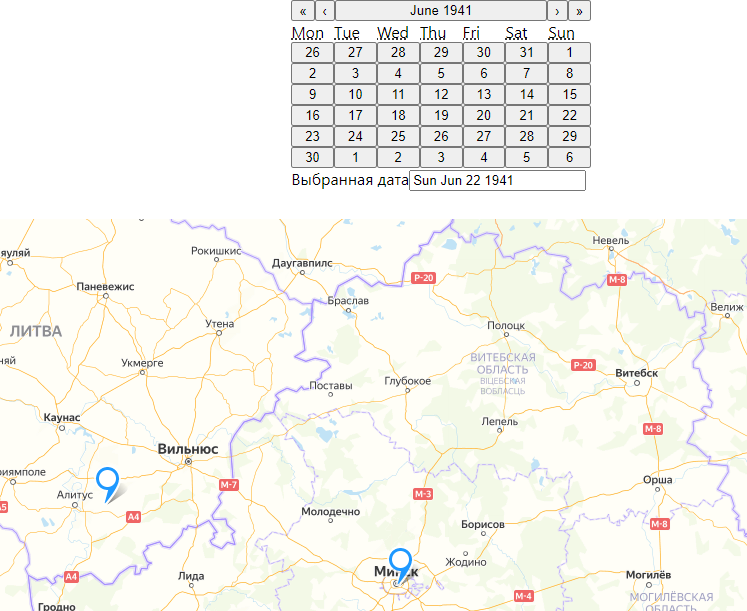


Рисунок 37 – отображение местоположений подразделений



Рисунок – отображение информации о подразделении

Заключение

В результате выполнения выпускной квалификационной работы было спроектировано и разработано приложение для визуализации данных.

На этапе проведения исследовательских работ был проведен анализ существующих систем; сформирован перечень задач, подлежащих решению в процессе разработки системы; обоснован выбор инструментов и платформы для разработки системы.

В результате выполнения проектно-конструкторских работ были решены основные архитектурные задачи, на основании которых было разработано веб-приложение, решающие поставленные задачи.

По итогам проведения проектно-технологических работ были реализованы frontend и backend приложения с учетом всех предъявляемых требований, функционал которых соответствует техническому заданию.

В качестве развития проекта можно выделить добавление функционала проигрывателя с возможностью установки скорости изменения даты и времени для автоматической смены отображаемых данных; навигации по ключевым событиям; отрисовки границ и площадей, занимаемых войсками; добавление анимации плавного изменения местоположения подразделения; добавление функционала обратной связи для получения представления об удобстве использования приложения, предложений о развитии системы.

Цель данной работы, состоящая в разработке и реализации frontend и backend приложения для визуализации данных о передвижениях войск Красной Армии во время Великой Отечественной войны 1941-1945 гг., была достигнута и поставленные при проектировании задачи решены в полном объеме.

Список использованных источников

1. Список общевойсковых армий вооружённых сил СССР (1941-1945) [Электронный ресурс] – https://ru.freejournal.org/3582244/1/spisok-obshchevoyskovykh-armiy-vooruzhennykh-sil-sssr-1941-1945.html . Дата обращения:7.01.2021.
2. Язык C# и платформа .Net Core [Электронный ресурс] –https://metanit.com/sharp/tutorial/1.1.php . Дата обращения: 10.01.2021.
3. Руководство по ASP.NET Core 5 [Электронный ресурс] – <https://metanit.com/sharp/aspnet5/> . Дата обращения: 18.01.2021.
4. Руководство по EntityFrameworkCore [Электронный ресурс] – <https://metanit.com/sharp/entityframeworkcore/> . Дата обращения: 25.01.2021.
5. ASP.NET Core Identity [Электронный ресурс] – <https://metanit.com/sharp/aspnet5/16.1.php> . Дата обращения: 02.02.2021.
6. Язык программирования Javascript:особенности и преимущества [Электронный ресурс] – <https://vc.ru/hr/145461-yazyk-programmirovaniya-javascript-osobennosti-i-preimushchestva>. Дата обращения: 09.02.2021.
7. Документация React [Электронный ресурс] – <https://reactjs.org> Дата обращения: 16.02.2021.
8. Документация material-ui [Электронный ресурс] – <https://material-ui.com/ru/> . Дата обращения: 23.02.2021.
9. Диаграмма сценариев использования в процессе разработки ПО [Электронный ресурс] – <https://habr.com/ru/company/luxoft/blog/312188/> . Дата обращения: 1.03.2021.
10. Обзор нотаций, используемых при построении диаграмм "сущность-связь" [Электронный ресурс] –<http://www.mstu.edu.ru/study/materials/zelenkov/ch_2_4.html> . Дата обращения: 08.03.2021
11. Архитектура клиент – сервер [Электронный ресурс] – <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-6-ch1-arhitektura-klient-server.html>. Дата обращения :15.03.2021
12. Что такое REST API ? [Электронный ресурс] – <https://starkovden.github.io/what-is-rest-api.html> . Дата обращения : 22.03.2021.

Приложение А

В графическую часть выпускной квалификационной работы входят

* Анализ аналогов
* Концептуальная модель предметной области
* Диаграмма вариантов использования
* Логическая схема базы дан
* ных
* Интерфейс взаимодействия незарегистрированного пользователя с системой
* Интерфейс взаимодействия зарегистрированного пользователя
* Схема взаимодействия пользователя с системой.

1. API Яндекс.Карт — это набор сервисов, которые позволяют использовать картографические данные и технологии Яндекса в собственных проектах. [↑](#footnote-ref-1)
2. [Cross-Origin Resource Sharing (CORS)](http://www.w3.org/TR/cors/) является техникой для ослабления правила одного источника, позволяя JavaScript на web странице обрабатывать ответ от REST API от другого источника. [↑](#footnote-ref-2)
3. Swagger является фреймворком для спецификации REST API. REST API представляет из себя набор соглашений, который необходимо реализовать для удобной работы с сущностями в условиях передачи данных. [↑](#footnote-ref-3)
4. Language Integrated Query (LINQ) — проект [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) по добавлению [синтаксиса](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81) языка запросов, напоминающего [SQL](https://ru.wikipedia.org/wiki/SQL), в языки программирования платформы [.NET Framework](https://ru.wikipedia.org/wiki/Microsoft_.NET). [↑](#footnote-ref-4)
5. В соответствии с объектной моделью документа («Document Object Model», коротко DOM), каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом. [↑](#footnote-ref-5)
6. Axios — это JavaScript-библиотека для выполнения либо HTTP-запросов в Node. js, либо XMLHttpRequests в браузере [↑](#footnote-ref-6)