|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУ-КФ «Информатика и управление»\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_ ***ИУ5-КФ «Системы обработки информации»***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

**«**Разработка веб-приложения онлайн-карты передвижения войск СССР во время великой отечественной войны 1941-1945 гг.»

по дисциплине ***Разработка программных систем***

Студент гр. САПР.Б-71 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Рахманкулов А.Р.)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Кириллов В. Ю.)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2020 г.

СОДЕРЖАНИЕ

[1 Техническое задание 4](#_Toc60242033)

[1.1 Введение 4](#_Toc60242034)

[1.1.1 Наименование программы 4](#_Toc60242035)

[1.1.2 Область применения 4](#_Toc60242036)

[1.1.3 Объект, в котором используют программу 4](#_Toc60242037)

[1.1.4 Основание для разработки 4](#_Toc60242038)

[1.2 Назначение разработки 4](#_Toc60242039)

[1.3 Требования к программе 5](#_Toc60242040)

[1.3.1 Требования к функциональным характеристикам 5](#_Toc60242041)

[1.3.2 Требования к надежности 5](#_Toc60242042)

[1.3.3 Условия эксплуатации 5](#_Toc60242043)

[1.3.4 Требования к составу и параметрам технических средств 5](#_Toc60242044)

[1.3.5 Требования информационной и программной совместимости 6](#_Toc60242045)

[1.4 Стадии и этапы разработки 6](#_Toc60242046)

[1.5 Порядок контроля и приёмки 7](#_Toc60242047)

[1.5.1 Методы испытания системы 7](#_Toc60242048)

[1.5.2 Общие требования к приёмке работ по стадиям 7](#_Toc60242049)

[1.5.3 Статус приемочной комиссии 7](#_Toc60242050)

[1.6 Требования к программной документации 7](#_Toc60242051)

[2 Исследовательская часть 8](#_Toc60242052)

[2.1 Постановка задачи проектирования 8](#_Toc60242053)

[2.2 Описание предметной области 8](#_Toc60242054)

[2.2.1 Общие сведения 8](#_Toc60242055)

[2.2.2 Семантика. Сущности и отношения 10](#_Toc60242056)

[2.2.3 Описание процессов предметной области 12](#_Toc60242057)

[2.3 Анализ аналогов 13](#_Toc60242058)

[2.3.1 Coronavirus-monitor.ru 13](#_Toc60242059)

[2.3.2 Видеоролик на Youtube 15](#_Toc60242060)

[2.3.3 Веб-сайт pobediteli.ru 17](#_Toc60242061)

[2.4 Перечень задач, подлежащих решению в процессе разработки 18](#_Toc60242062)

[2.5 Обоснование выбора инструментов и платформы для разработки 18](#_Toc60242063)

[2.5.1 С# и .Net 18](#_Toc60242064)

[2.5.2 JavaScript 25](#_Toc60242065)

[2.5.3 React. Преимущества и недостатки 32](#_Toc60242066)

[3 Проектно-технологическая часть 34](#_Toc60242067)

[3.1 Проектирование начального и тестового наполнения базы данных 34](#_Toc60242068)

[4 Проектно-конструкторская часть 37](#_Toc60242069)

[4.1 Разработка структуры приложения. 37](#_Toc60242070)

[4.2 Логическая структура базы данных 38](#_Toc60242071)

[4.3 Разработка интерфейса взаимодействия пользователя с системой 39](#_Toc60242072)

[4.4 Разработка архитектуры приложения. 40](#_Toc60242073)

[4.5 Реализация функционирующего приложения 44](#_Toc60242074)

[4.5.1 Серверный компонент 44](#_Toc60242075)

[4.5.2 Клиентское приложение 49](#_Toc60242076)

[5 Список литературы 55](#_Toc60242077)

# Техническое задание

## Введение

### Наименование программы

Разработка веб-приложения онлайн-карты передвижения войск СССР 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 СССР в период 1941-1945 гг. В результате, появится наглядное представление о передвижении армейских подразделений во время войны.

### 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 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.

Дата утверждения документа: \_\_.09.2020.

## Назначение разработки

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

Главной задачей создания данной карты является отображение исторической действительности исходя из архивов.

## Требования к программе

### Требования к функциональным характеристикам

Приложение должно выполнять следующие функции:

Информационное обслуживание: отображение информации в зависимости от заданной даты; отображение сведений о подразделениях.

### Требования к надежности

Сервер должен обеспечивать устойчивое функционирование сайта:

* обеспечение работоспособности при больших нагрузках,
* работоспособность сервера при вводе 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 клиентской части приложения;
* написание серверной части;
* сопровождение приложения.

### Требования к составу и параметрам технических средств

Запуск разработанного Web-приложения и надёжность его работы без сбоев должно обеспечивать аппаратное обеспечение.

Технические характеристики компьютера, 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

#### Требования к компоновке страниц веб-приложения

Приложение должно обеспечивать адаптивный дизайн при переходе на различные устройства.

#### Требования к функциональности веб-сервера

Веб-сервер должен обеспечивать возможности:

* управления содержанием и структурой передаваемых данных.

## Стадии и этапы разработки

Первый этап – изучение предметной области, определение состава и структуры данных. В результате прохождения первого этапа определяются: логическая структура данных, ТЗ, описание целей разработки.

Второй этап – анализ пользовательских сценариев работы. В результате создаётся пользовательский интерфейс.

Третий этап – анализ взаимодействия веб-приложения с серверным компонентом. В результате определяются: архитектура приложения и структура классов.

## Порядок контроля и приёмки

### Методы испытания системы

Проверка работоспособности сайта на различных веб-браузерах (Mozila, Chrome, Internet Explorer).

### Общие требования к приёмке работ по стадиям

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

### Статус приемочной комиссии

Прием работы на всех этапах осуществляется непосредственным руководителем курсовой работы. По окончании разработки работоспособное приложение демонстрируется на комиссии.

## Требования к программной документации

- техническое задание по разрабатываемой системе;

- расчётно-пояснительная записка по разработанной системе;

- сопровождающая презентация в формате .ppt или .pptx.

# Исследовательская часть

## Постановка задачи проектирования

Целью данной курсовой работы является проектирование прототипов серверной и клиентской части веб-приложения для визуализации данных.

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. Выделить этапы проектирования веб-приложения.
5. Разработать веб-приложение.
6. Протестировать и выделить преимущества и недостатки веб-приложения.

## Описание предметной области

### Общие сведения

Великая Отечественная война - одно из немногих исторических событий, активно присутствующих в коллективной памяти россиян и респондентов стран СНГ. Данные социологических опросов показывают, что около 60% россиян считают Победу 9-го мая самой большой победой России за всю ее историю. И все же, по мнению 40% опрошенных, новые поколения начинают забывать об этой странице прошлого. Похожая тенденция, но в гораздо большей степени, уже давно наблюдается и в странах Европы, где в последние годы происходит и спад интереса ко Второй мировой войне, и изменение акцентов в ее оценке.

Левада-центр провел опрос об интересе граждан к историческим эпохам и наиболее значимым, по мнению респондентов, периодам в истории России. По-прежнему больше всего «особый, повышенный интерес» у респондентов вызывает Великая Отечественная война. Но если в июле 2008 года к этому периоду внимание проявляли более половины респондентов (55%), то в марте 2017 года — всего 38%.

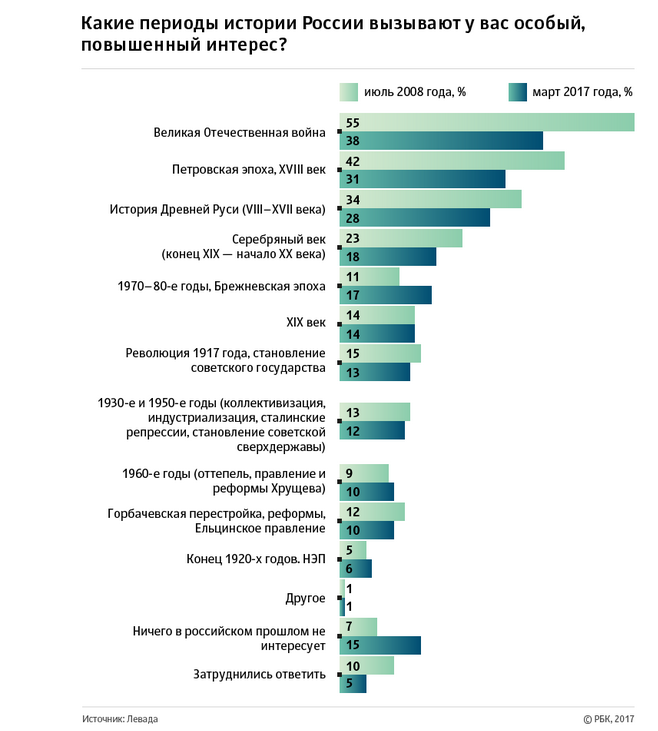


Рисунок 1– данные опроса

«Эта тема властями продвигается из-за отсутствия других столь значимых побед, с уходом ветеранов появился «Бессмертный полк». Однако Великой Отечественной интересуются уже менее половины россиян, ведь в силу давности событий война переходит в область мифа и вытесняется из массового сознания», — рассуждает эксперт Николай Миронов.

«Великая Отечественная война остается для россиян важным историческим событием в жизни страны, но за последние годы общий интерес к истории упал», — заметил Алексей Гражданкин (замдиректор Левада-Центра). По его мнению, общий спад интереса к истории можно сравнить с послеперестроечной эпохой 1990-х годов. Тогда из-за «лавины информации, обрушившейся на людей в перестройку», граждане стали меньше реагировать на исторические темы.

Социолог также указывает, что интерес к истории зависит от возраста. Молодежь в принципе меньше пожилых интересуется историей. Кроме того, молодые респонденты (представители «путинского поколения») больше внимания проявляют к истории Киевской Руси, в то время как брежневская эпоха интересует людей пенсионного и предпенсионного возраста. История Великой Отечественной интересует как людей старшего поколения, так и молодежь. Несколько меньше к ней проявляют внимание люди около 40 лет, социализировавшиеся в 1990-е годы, замечает Гражданкин. «Это отражает повестку тех лет», — считает социолог.

Опрос Левада-центра был проведен 2–6 марта 2017 года по репрезента​тивной всероссийской выборке городского и сельского населения среди 1,6 тыс. человек. Опрашивались люди в возрасте 18 лет и старше в 137 населенных пунктах 48 регионов страны. Респондентов лично интервьюировали на дому.

### Семантика. Сущности и отношения

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.

Отдельные (ОА), ударные (УдА), гвардейские армии (Гв.А) и Армии фронтов РККА во время Великой Отечественной войны (Второй мировой войны) чаще всего предназначались для выполнения оперативных задач в составе фронтов, а иногда и самостоятельно. В их состав входили управление, несколько соединений и отдельных частей различных родов войск (сил), специальных войск и служб.

За время ВОв организация армии претерпела ряд изменений. До неё управление армии, численностью 285 человек при 25 легковых, 2 грузовых автомобилях и 4 автобусах, содержалось по штату № 2/12 от 13 сентября 1940 года. В армии имелись командующий войсками армии, его заместители, военный совет и штаб, отделы: политической пропаганды, военно-воздушных сил (ВВС), боевой подготовки, кадров, противовоздушной обороны (ПВО), артиллерийский, автобронетанковый, инженерный, связи, химический, снабжения горючим, интендантский, санитарный, ветеринарный, финансовый, а также подразделения обслуживания. В целом, такой же принцип организации управления армии и сохранился в войну.

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. В состав армии входила и авиация в виде авиационных дивизий и полков. После начала войны из-за трудностей, связанных с управлением и больших потерь, армии уменьшились и в основном стали представлять собой совокупность 5 — 6 стрелковых дивизий, не объединённых в корпуса. Поздней осенью 1941 года некоторые армии формировались даже не из дивизий, а из стрелковых бригад. В 1942 году в ВС СССР начали процесс постепенного восстановления армий: в ней появилось больше различного вооружения, техники, увеличилось количество отдельных, в том числе, специальных, частей, и корпусное звено было восстановлено. Весной 1942 года армии остались без собственной авиации (за редкими незначимыми исключениями); авиация была выведена из армейского комплекта и сведена в воздушные армии РККА. Увеличивалось количество артиллерии: по общему правилу в армии имелись армейские пушечный, истребительно-противотанковый, зенитный, артиллерийский и миномётный полки (армейская артиллерия). В 1944 году количество вооружения позволило включать в состав армии пушечную артиллерийскую бригаду и отдельный танковый полк. Со второй половины 1944 года армия в основном представляла собой 3 — 4 стрелковых корпуса (7 — 12 дивизий), 3 — 4 артиллерийских и миномётных полка или артиллерийскую бригаду, отдельный танковый полк и части специального назначения.

Армия имела свой номер, исключая Приморскую, которая в двух формированиях имела только территориальное наименование, а также Ленинградскую армию народного ополчения. Предназначенные для наступления на направлении главного удара фронта, наиболее мощные по боевому составу армии осенью 1941 года получили наименование «ударных».

В результате анализа предметной области были сформированы сущности:

* Подразделение;
* Местоположение;
* Актуальные данные
* Командир
* Звание
* Тип подразделения
* Удостоверяющий документ

Концептуальная модель предметной области изображена в «Приложении 1».

### Описание процессов предметной области

Попадая на сайт приложения пользователю отрисовывается на карте расположение подразделений на начало военных действий (22.06.1941).

Пользователю будет предоставлена возможность выбора даты. Также, для определения конкретного подразделения необходимо выводить информацию о нём. В этом и будет заключаться интерактивность.

## Анализ аналогов

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

В качестве аналогов выделим:

1. Онлайн карту распространения коронавирусной инфекции по миру на сайте coronavirus-monitor.ru
2. Видеоролики по запросу «вторая мировая война на карте» на видеохостинге Youtube.
3. Веб сайт pobediteli.ru

### Coronavirus-monitor.ru

В качестве примера использования карты выделим сайт coronavirus-monitor.ru

Простой дизайн позволит быстро разобраться с возможностями.

Преимущества:

1.Присутствует дополнительная статистическая информация

2. Возможна фильтрация данных (только Россия).

3.Адаптивный дизайн (BlueStacks v 4,Android 7)

4.Присутствуют комментарии, где можно свободно указать пожелания и принять, хоть и косвенно, участие в разработке приложения

Недостатки:

1. Навязчивая реклама занимающая часть полезного контента.

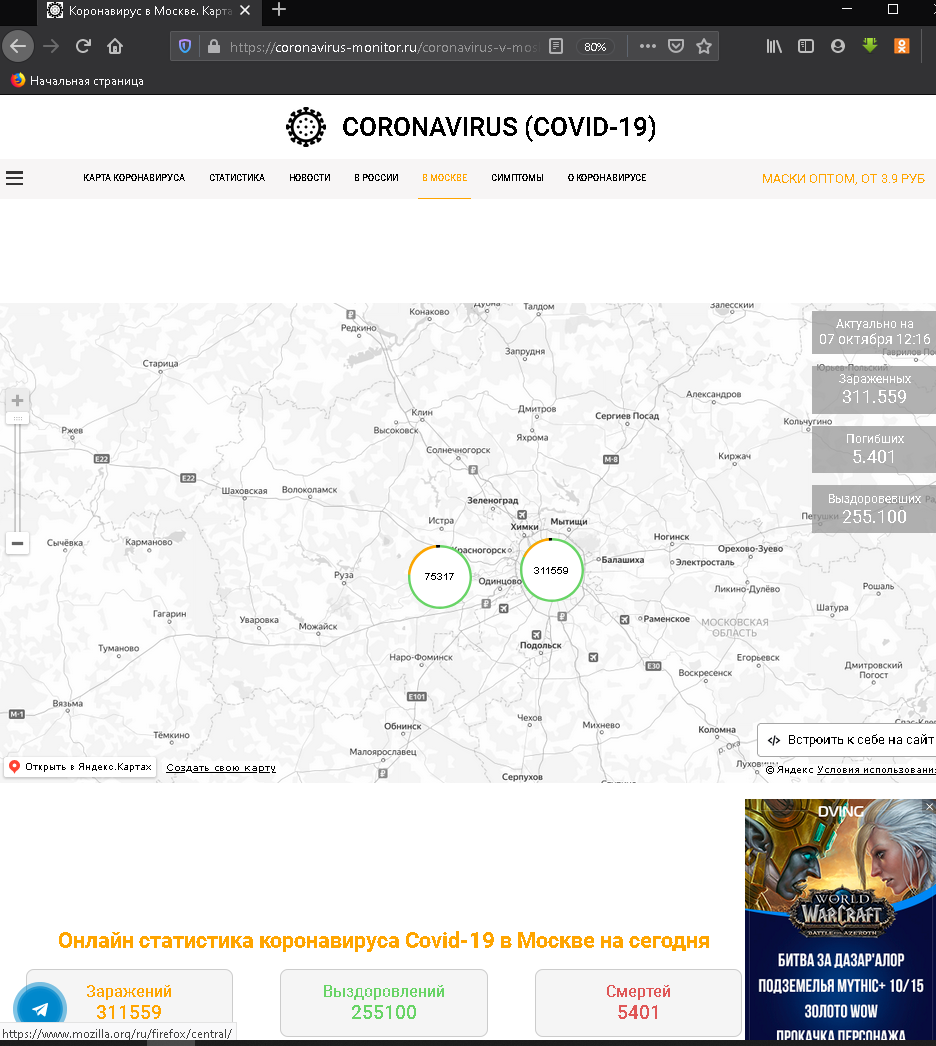


Рисунок - сайт coronavirus.monitor.ru

### 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.

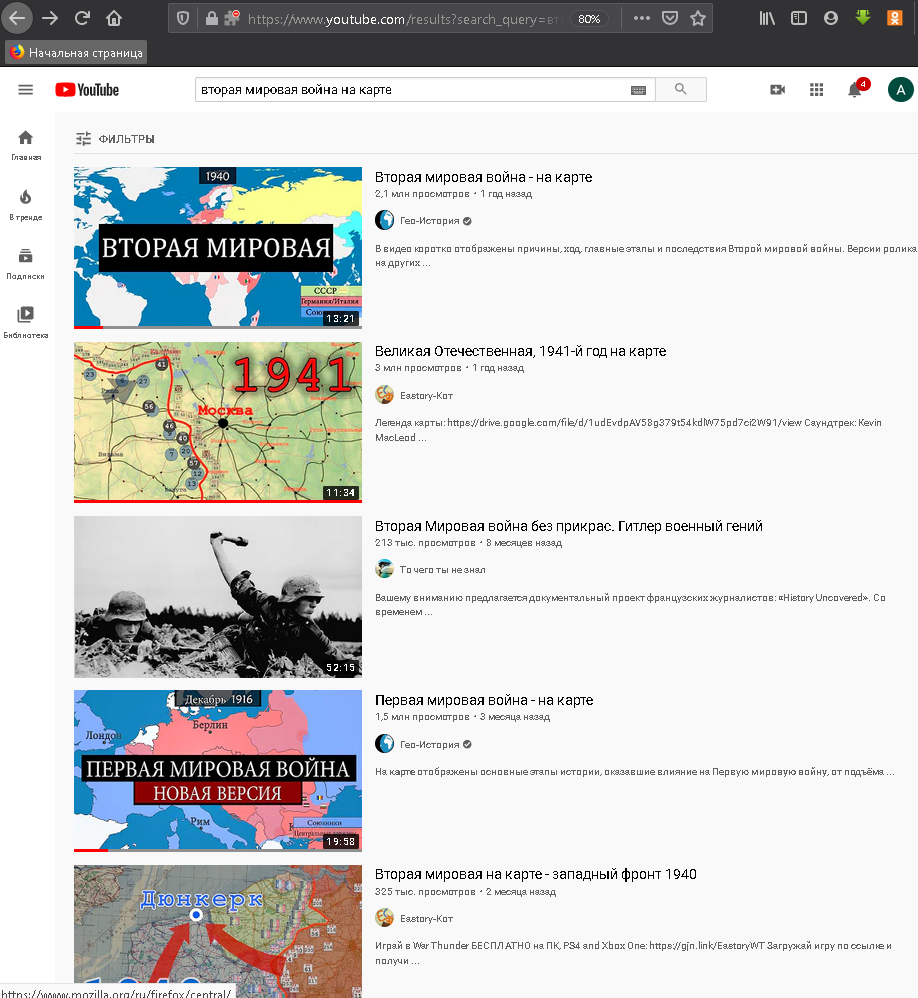


Рисунок – запрос на хостинге Youtube

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>

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:

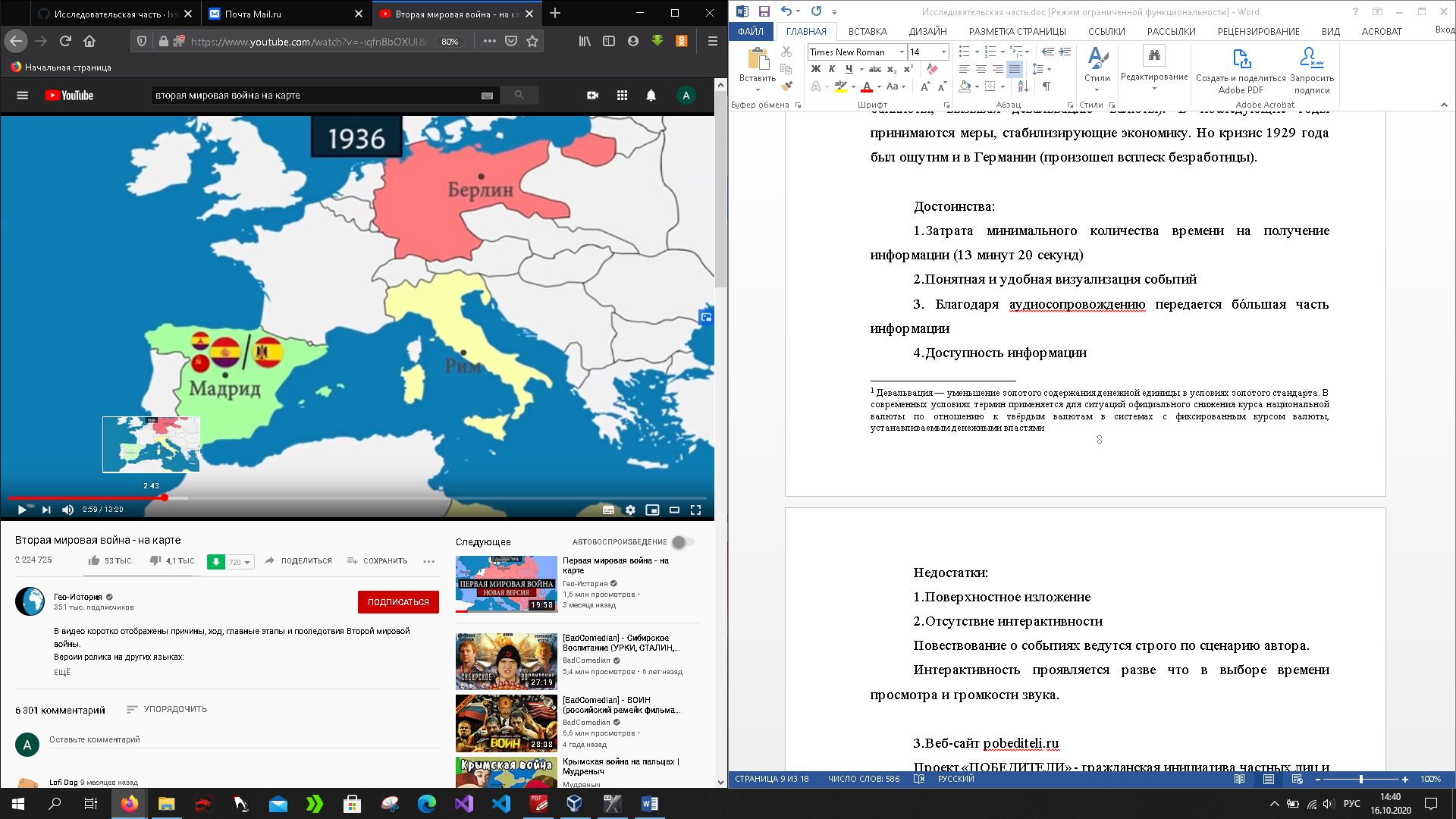
1.Слишком поверхностное изложение

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

2.Отсутствие интерактивности

Повествования о событиях ведутся строго по сценарию автора.

Интерактивность проявляется разве что в выборе времени просмотра и громкости звука.



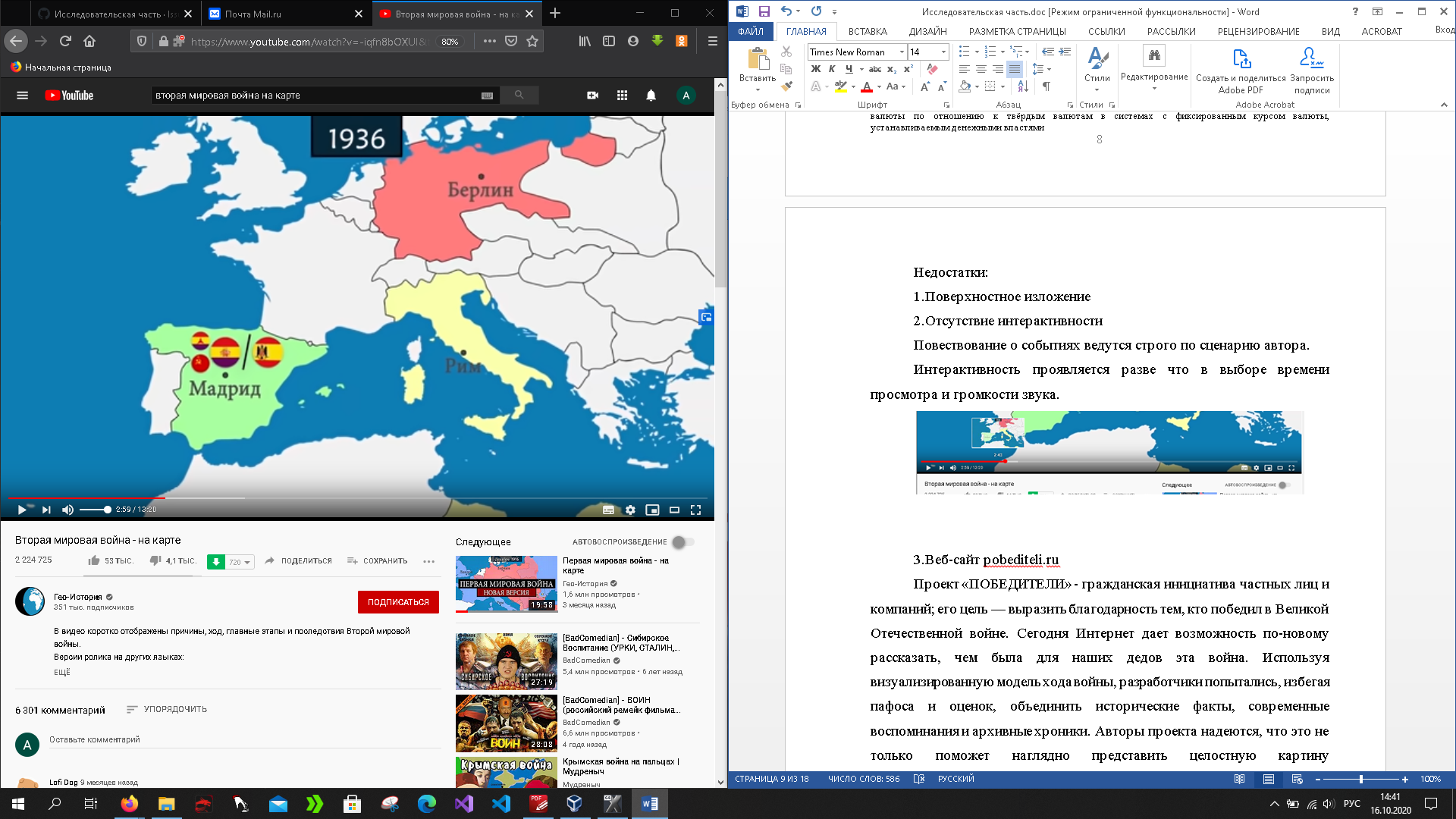


Рисунок 4 – интерактивность

3.Есть возможность столкнуться с ненормативной лексикой в комментариях

В комментариях под видео пользователи имеют возможность высказаться относительно просмотренного контента. Человек может не согласиться с точкой зрения автора и начинает излагать своё видение. Но может найтись и третий, который против всех. Это может перерасти в спор. Порой, люди слишком увлекаются и могут даже перейти на личности.

### 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 дедов 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.

Преимущества:

Исчерпывающая информация об исторических событиях

Можно ознакомиться с официальными документами не тратя времени на их поиск в сети интернет.

Также в проекте присутствует возможность прослушивания комментариев очевидцев.

1. Удобство использования:

- В приложении присутствует справка о функциональности

- Возможна навигация по дате и связанными с ней историческими собитиями

- Возможна навигация по дате и связанными с ней историческими собитиями

- Информация черпается из официальных источников

- Присутствует возможность скачивания программы

- Достаточно исчерпывающая информация о самом проекте

- Есть возможность оставить отзыв

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.Несмотря на то, что в списках присутствует один немецкий ветеран, можно сказать, что информация о них отсутствует.

## Перечень задач, подлежащих решению в процессе разработки

Для создания системы необходимо:

* Определить состав и структуру данных;
* Определить архитектуру приложения и структуры классов;
* Спроектировать пользовательский интерфейс;
* Спроектировать серверный компонент.

## Обоснование выбора инструментов и платформы для разработки

### С# и .Net

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 программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

#### История развития языка

Язык программирования С# был разработан в 1993—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft.



Рисунок 5 – Андрес Хейлсберг

К 2000 году у Microsoft были готовы промышленные версии новых технологий и решений для обмена сообщениями и данными, а также для создания Internet-приложений. Была выпущена и новая платформа для разработки под новые решения — .NET. В ней объединились сразу несколько языков программирования, что было в новинку для того времени.

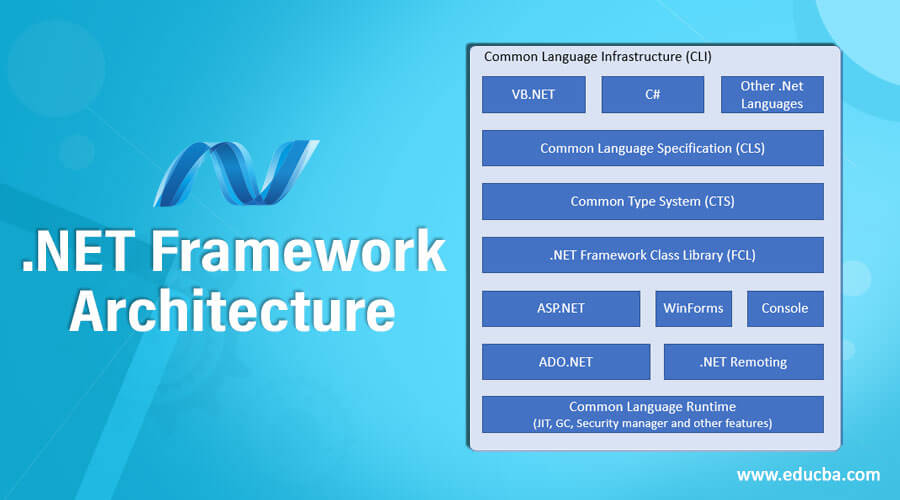


Рисунок 6 – архитектура .Net Framework

Ещё одним новшеством платформы .NET была технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно было относительно быстро разработать веб-приложения, взаимодействующие с базами данных. Специально для ASP.NET был создан язык программирования C#. Да и сама ASP.NET была полностью написана на нём.

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# 8.0, которая вышла в сентябре 2019 года вместе с релизом .NET Core 3.

C# — это язык с C-подобным синтаксисом. Здесь он близок в этом отношении к C++ и Java.

Будучи объектно-ориентированным языком, он много перенял у Java и С++. Как и Java, C# изначально предназначался для веб-разработки, и примерно 75% его синтаксических возможностей такие же, как у  Java. C# также называют «очищенной версией Java». Ещё 10% наш герой позаимствовал из C++ и 5% – из Visual Basic. Оставшиеся 10% C# — это реализация собственных идей разработчиков. 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.

#### Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

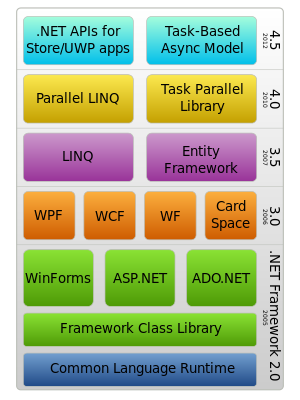


Рисунок 7– технологии .Net Framework

Когда-то Билл Гейтс сказал, что .NET — это лучшее, что сделала компания Microsoft. У него есть весомые основания так считать. Фреймворк .NET представляет мощную платформу для создания приложений. Вот его киллер-фичи:

* Поддержка нескольких языков. В основе  .NET — общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему платформа поддерживает несколько языков: наряду с C# это VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. Код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) — своего рода ассемблер платформы .NET. Поэтому можно сделать отдельные модули одного приложения на разных языках.
* Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. Какое бы приложение мы ни собирались писать на C# — текстовый редактор, чат или сложный веб-сайт — так или иначе мы задействуем библиотеку классов .NET.
* Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов — это основа для целого стека технологий, которые разработчики могут задействовать при создании разных приложений. Например, для баз данных в этом стеке имеется технология ADO.NET и Entity Framework Core. Для графических приложений с насыщенным интерфейсом — технологии WPF и UWP. Для более простых графических приложений — Windows Forms. Для разработки мобильных приложений — Xamarin. Для создания веб-сайтов - ASP.NET и т.д.

#### .NET Framework, .NET Core и Xamarin

.NET долгое время развивался под названием .NET Framework — преимущественно как платформа для Windows. Но с 2019 она больше не развивается — последней версией этой платформы стала .NET Framework 4.8.

В 2014 Microsoft начал выпускать альтернативную платформу - .NET Core, которая должна была вобрать в себя все возможности устаревшего .NET Framework и добавить новую функциональность. Поэтому следует различать .NET Framework, который предназначен преимущественно для Windows, и кроссплатформенный .NET Core.

Xamarin — это платформа с открытым исходным кодом, предназначенная для построения современных производительных приложений для iOS, Android и Windows с .NET.

Платформа Xamarin ориентирована на разработчиков, перед которыми стоят следующие задачи:

* Совместное использование кода, тестов и бизнес-логики на различных платформах.
* Написание кроссплатформенных приложений на языке C# в Visual Studio.

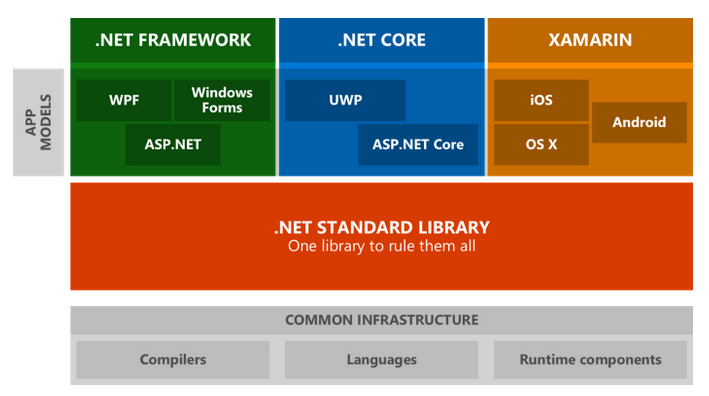


Рисунок 8 – Архитектура .Net

#### Преимущества и недостатки языка C#

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.
* Большое количество «синтаксического «сахара» — специальных конструкций, разработанных для понимания и написания кода. Они не имеют значения при компиляции.
* 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.

### JavaScript

#### История JavaScript

JavaScript существует уже четверть века. Срок внушительный для подобных технологий. За это время он прошёл большой путь, стал мощным инструментом IT-индустрии. По уровню популярности язык в группе лидеров. При этом он считается доступным для освоения, дружественным по отношению к новичкам.

Само наименование Javascript появилось не сразу. Изначально язык назывался Mocha, затем появился термин Livescript. Это гораздо более ёмко отражало его суть. В то время в браузерах активно внедрялась поддержка нового, перспективного языка Java. Тогда и было решено переименовать Livescript в Javascript. Схожесть между языками есть, прежде всего, по синтаксису. Но по сути это очень разные инструменты.

В 2015 году вышла версия ES6 (ES2015). Это событие считается прорывом в развитии Javascript. Появились новые стандарты и возможности. Например – константы.

Код стал более рациональным, была реализована идея «пиши меньше – делай больше».

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.

#### Востребованность, сферы применения

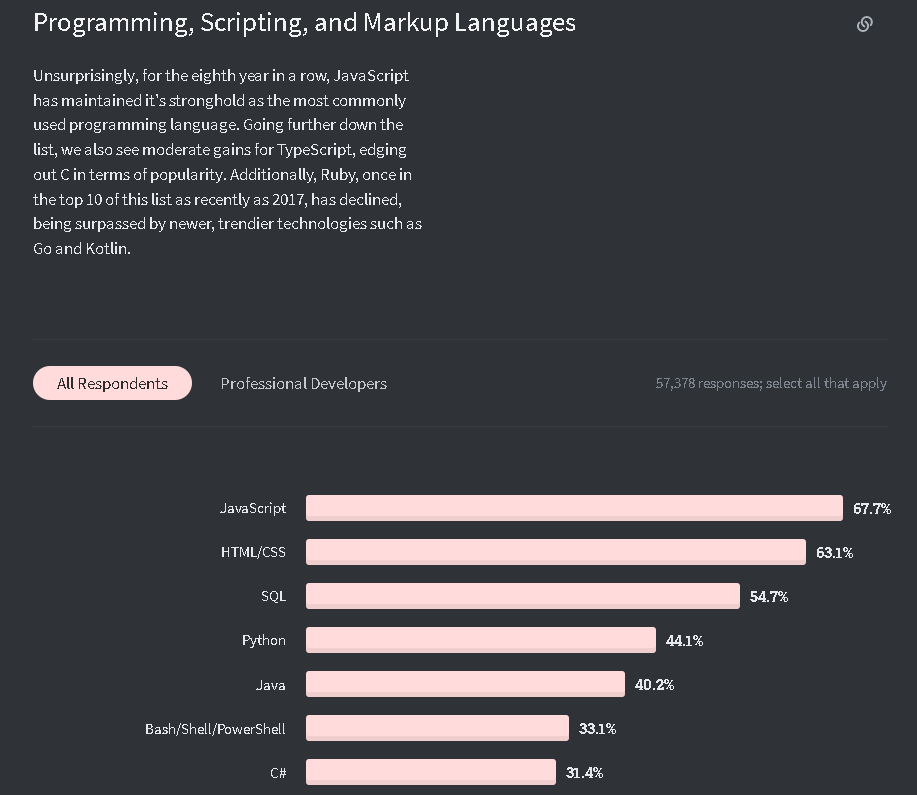
Неизменно высокий интерес к Javascript подтверждают специальные международные рейтинги. Эти индексы представляют собой аналитические данные, в основе которых:

* запросы по языкам программирования в поисковых системах;
* вопросы на профессиональных форумах;
* проекты, реализованные с использованием тех или иных технологий;
* вакансии для программистов от крупнейших рекрутинговых агентств.

**PYPL (PopularitY of Programming Language)**. Авторитетный рейтинг, в его основе – данные крупнейшего поисковика Google SE. Javascript – в тройке лидеров.

Рисунок 9 – популярность языков программирования согласто PYPL

**Stack Overflow**. Этот индекс анализирует количество вопросов по языкам и инструментам в профессиональных сообществах. Здесь Javascript уверенно занимает лидирующую позицию.

Рисунок 10 – популярность языков программирования согласто Stack Overflow

**W3Techs - World Wide Web Technology Surveys**. Основа показателей – количество веб-проектов, использующих те или иные технологии. Выбираем раздел клиентских интерфейсов. Видим, что в нём у Javascript конкурентов нет, это основной инструмент.

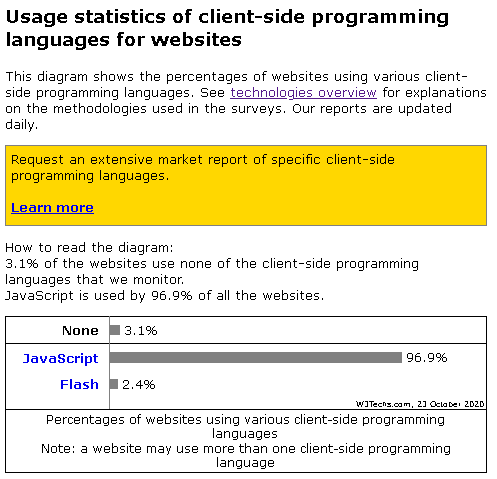


Рисунок 11 – популярность языков программирования согласто W3Techs

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. Самые популярные из них, пожалуй, следующие:

Angular JS - это фреймворк для работы с JavaScript, поддерживаемый Google. Он используется для создания одностраничных приложений и позволяет взаимодействовать с DOM (англ. Document Object Model — объектная модель документа).

Meteor.js - веб-платформа на языке [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), предназначенная для разработки Web-приложений [реального времени](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F). Для связи с современными [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) используется протокол [Distributed Data Protocol](https://ru.wikipedia.org/wiki/Distributed_Data_Protocol) (DDP), поддерживаемый с помощью [WebSocket](https://ru.wikipedia.org/wiki/WebSocket)'ов, либо, если поддержки веб-сокетов и DDP нет — [AJAX](https://ru.wikipedia.org/wiki/AJAX).

Node.js - программная платформа, основанная на движке [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) (транслирующем [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) в [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами [ввода-вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) через свой [API](https://ru.wikipedia.org/wiki/API) (написанный на [C++](https://ru.wikipedia.org/wiki/C%2B%2B)), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль [веб-сервера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи [NW.js](https://ru.wikipedia.org/wiki/NW.js), AppJS или [Electron](https://ru.wikipedia.org/wiki/Electron) для [Linux](https://ru.wikipedia.org/wiki/Linux), [Windows](https://ru.wikipedia.org/wiki/Windows) и [macOS](https://ru.wikipedia.org/wiki/MacOS)) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит [событийно-ориентированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [асинхронное (или реактивное)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирование с [неблокирующим вводом/выводом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B0%D1%8F_%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)

Aurelia.js - это модульный современный интерфейс Javascript для создания браузеров, мобильных и настольных приложений, основанных на открытых веб-стандартах. Вместо того, чтобы быть всеобъемлющей основой, Aurelia применяет подход к архитектуре для каждого модуля. Любая часть Aurelia может быть заменена третьей стороной (или вообще удалена)

Backbone.js - [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)-библиотека, основанная на шаблоне проектирования [Model-View-Presenter](https://ru.wikipedia.org/wiki/Model-View-Presenter) (MVP), предназначена для разработки веб-приложений с поддержкой [RESTful](https://ru.wikipedia.org/wiki/REST) [JSON](https://ru.wikipedia.org/wiki/JSON) интерфейса.

PureMVC - это компактный [прикладной фреймворк](http://ru.wikipedia.org/wiki/Framework), позволяющий разместить множество задач, разбитых на 3 составляющих, со слабой связанностью и удобным движком событий.

Ember.js — [свободный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%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) [каркас веб-приложений](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BA%D0%B0%D1%81_%D0%B2%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) на [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), реализующий шаблон [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller) и нацеленный на упрощение создания масштабируемых [одностраничных веб-приложений](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).

React.js - [JavaScript](https://ru.wikipedia.org/wiki/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) 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). React разрабатывается и поддерживается [Facebook](https://ru.wikipedia.org/wiki/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) и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость

jQuery - [набор функций JavaScript](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), фокусирующийся на взаимодействии [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) и [HTML](https://ru.wikipedia.org/wiki/HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими. Также библиотека jQuery предоставляет удобный [API](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%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX).

Vue.js - [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)-[фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) 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) для создания [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). Легко интегрируется в проекты с использованием других [JavaScript-библиотек](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). Может функционировать как [веб-фреймворк](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BA%D0%B0%D1%81_%D0%B2%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для разработки [одностраничных приложений](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) в [реактивном](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) стиле.

Javascript применяют также для создания мобильных приложений, в серверной (backend) разработке, в десктопных (например, офисных) программах. Мировой бум сайтостроения и тенденция к расширению интерактивности способствуют феноменальному росту востребованности языка.

#### Достоинства и недостатки

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.

— Мощная инфраструктура (экосистема). Первые 10 лет этого не было и в помине. Затем количество готовых решений в открытом доступе так возросло, что работать с Javascript и его фреймворками стало удовольствием.

— Простота и рациональность применения. Простую задачку можно решить за 5 минут, не надо делать лишнюю работу. Для сложных заданий есть варианты решения, можно подобрать лучший, адаптировать.

— Удобство пользовательских интерфейсов. Заполнение форм, выбор действий, активация кнопок, проверки ввода, реагирование на наведение / клики мыши и т.п

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

Как любой язык программирования, Javascript имеет некоторые недостатки (ограничения).

— 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. Выход HTML 5 дал новый толчок к расширению возможностей скриптов.

### React. Преимущества и недостатки

React - это декларативная, эффективная и гибкая библиотека JavaScript для создания пользовательских интерфейсов (UI). Она позволяет создавать сложные UI из небольших и изолированных частей кода, называемых «компонентами». Этот фреймворк отлично подходит для создания огромных веб-приложений, где данные могут меняться на регулярной основе.

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.
* Высокий уровень гибкости и максимальная отзывчивость.
* 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.
* В сочетании с ES6/7 ReactJS может легко работать при высоких нагрузках.
* Связывание данных от больших к меньшим. Это означает такой поток данных, при котором дочерние элементы не могут влиять на родительские данные.
* 100%-ая JavaScript-библиотека с открытым исходным кодом, которая получает множество ежедневных обновлений и улучшений в соответствии с отзывами разработчиков по всему миру.
* Невероятно легкий вес, так как данные, которые выполняются на стороне пользователя, могут легко быть представлены на стороне сервера в то же самое время.
* Миграция между версиями, как правило, очень проста. Также Facebook предоставляет «codemods» для автоматизации большей части этого процесса.

**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;
* Долгое время для освоения. React требует глубокого понимания того, как интегрировать пользовательский интерфейс в структуру MVC.

Компании, которые используют React: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft.

# Проектно-технологическая часть

## Проектирование начального и тестового наполнения базы данных

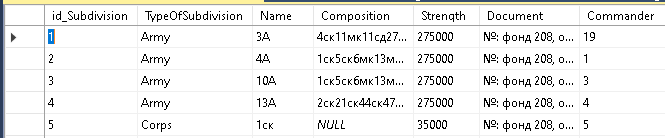


Рисунок 12 – тестовое наполнение данными

В качестве модели данных выступает сущность «Подразделение» с полями:

- Идентификатор подразделения

- Тип подразделения

- Название подразделения

- Состав

- Численность

- Удостоверяющий документ

- Командир

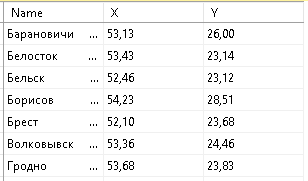


Рисунок 13–местоположение

Для задания местоположения подразделения на карте, необходимо иметь координаты городов. Для этого была введена сущность «Местоположение» с атрибутами:

- Название города;

- Координата X на карте;

- Координата Y на карте.

Для определения координат города был использован сервис yandex-map-constructor-location-tools.

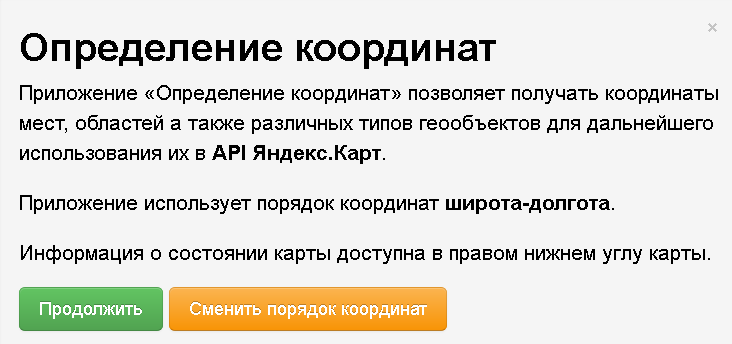


Рисунок 14 – начальная страница сервиса

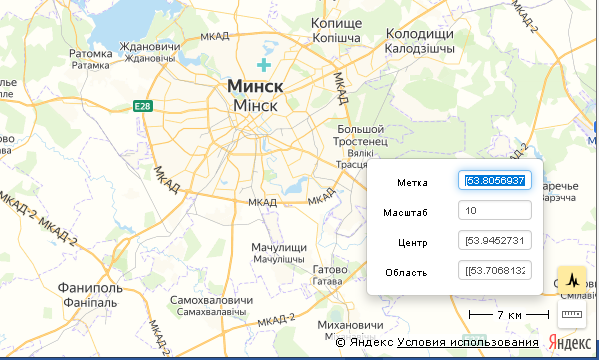


Рисунок 15 – пример определения координат

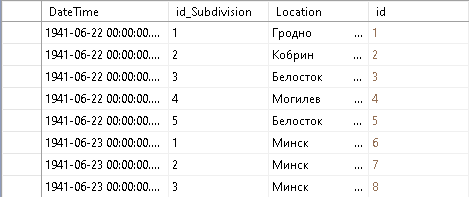


Рисунок 16 – актуальные данные

В качестве модели данных выступает сущность «Актуальные данные» с полями:

- Дата

- Идентификатор подразделения

- Местоположение

- Идентификатор данных

# Проектно-конструкторская часть

## Разработка структуры приложения.

В состав компонентов системы входят:

– Источник данных;

– Клиентское приложение для отображения данных;

– Серверная компонента для обработки исходных данных.

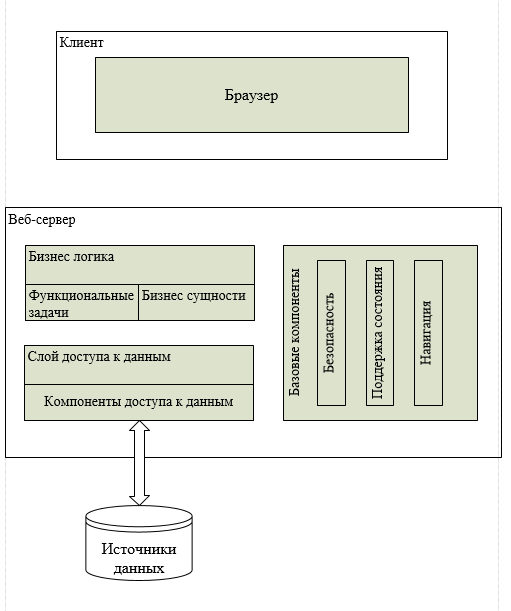


Рисунок 17 – структурная схема приложения

## Логическая структура базы данных

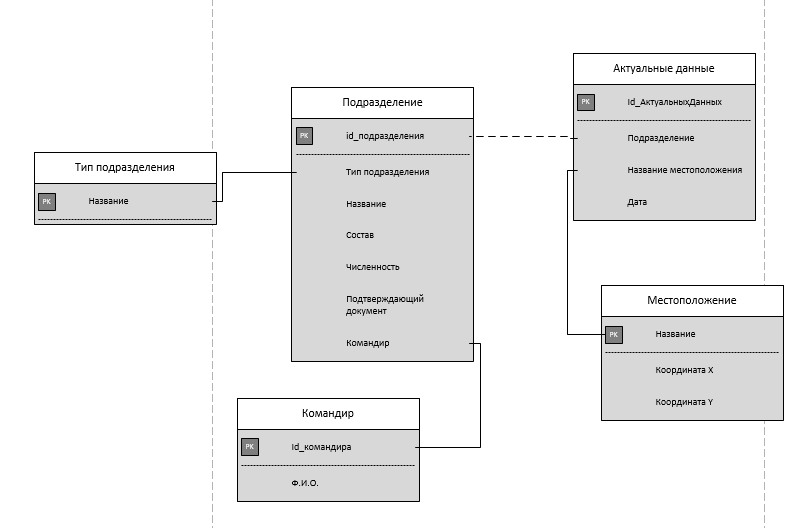


Рисунок 18– Концептуальная модель базы данных

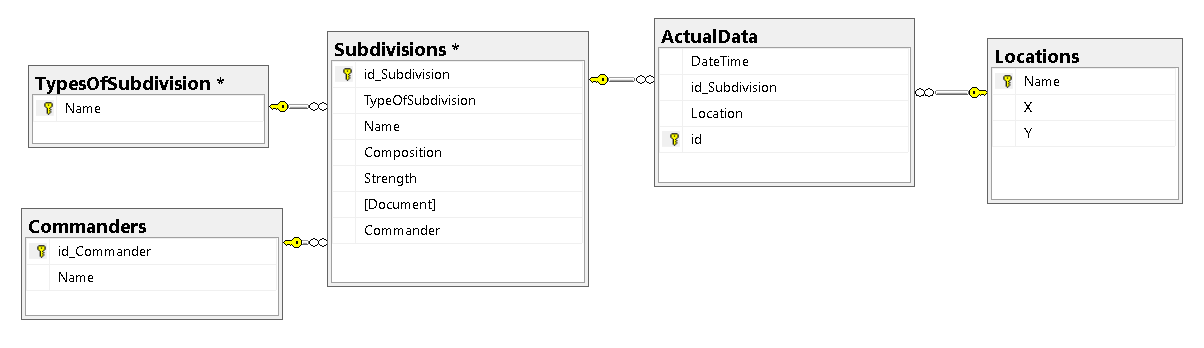


Рисунок 19 – логическая схема базы данных

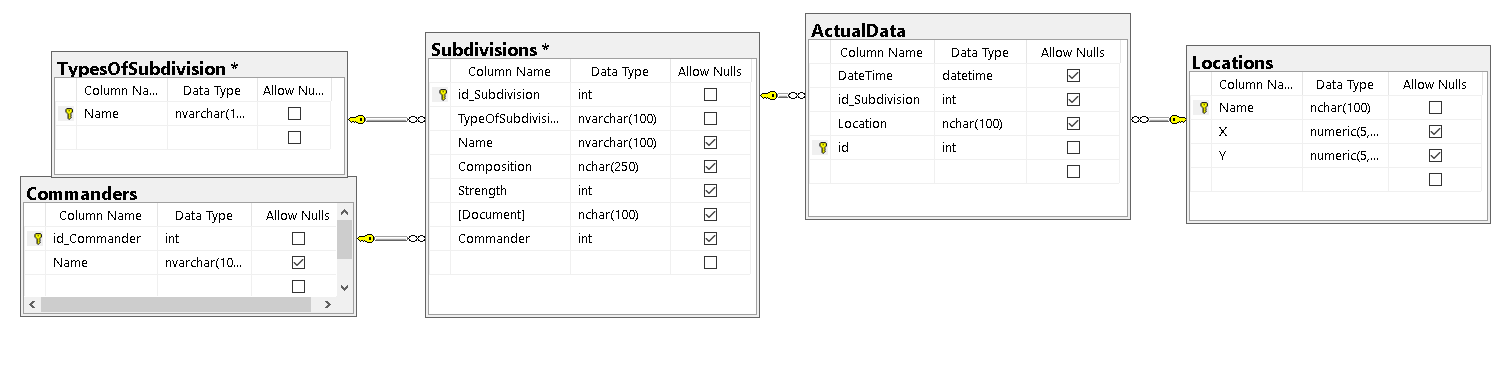


Рисунок 20– Физическая модель базы данных

## Разработка интерфейса взаимодействия пользователя с системой

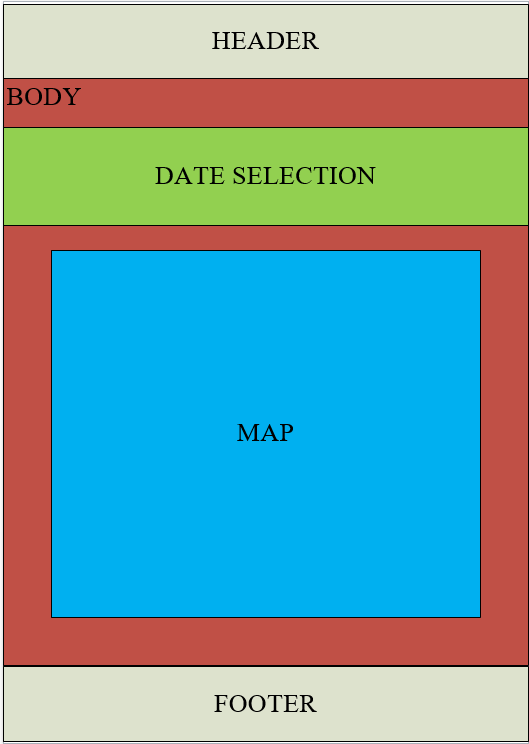


Рисунок 21– прототип интерфейса

## Разработка архитектуры приложения.

.NET — это бесплатная платформа разработки с открытым исходным кодом для создания различных типов приложений, таких как следующие:

* [Веб-приложения, веб-API и микрослужбы](https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core#recommended-learning-path)
* [Бессерверные функции в облаке](https://docs.microsoft.com/ru-ru/azure/azure-functions/functions-create-first-function-vs-code?pivots=programming-language-csharp)
* [Полностью облачные приложения](https://docs.microsoft.com/ru-ru/dotnet/architecture/cloud-native/)
* [Мобильные приложения.](https://dotnet.microsoft.com/learn/xamarin/hello-world-tutorial/intro)
* Классические приложения
* [Windows WPF](https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/)
* [Windows Forms](https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/)
* [Универсальная платформа Windows (UWP)](https://docs.microsoft.com/ru-ru/windows/uwp/get-started/create-a-hello-world-app-xaml-universal)
* [Игры](https://dotnet.microsoft.com/apps/games)
* [Интернет вещей.](https://docs.microsoft.com/ru-ru/dotnet/iot)
* [Машинное обучение](https://docs.microsoft.com/ru-ru/dotnet/machine-learning/)
* [Консольные приложения](https://docs.microsoft.com/ru-ru/dotnet/core/tutorials/with-visual-studio-code)
* [Службы Windows](https://docs.microsoft.com/ru-ru/aspnet/core/host-and-deploy/windows-service)

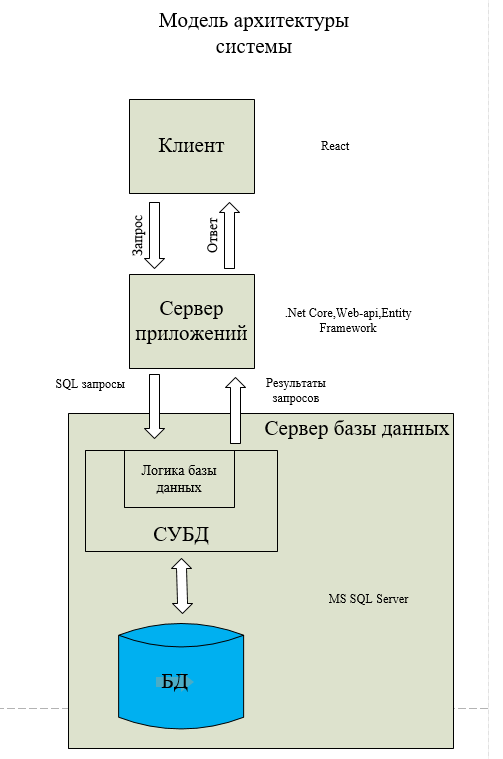


Рисунок 22 – Архитектура приложения

Сегодня большое значение имеет работа с данными. Для хранения данных используются различные системы управления базами данных: MS SQL Server, Oracle, MySQL и так далее. И большинство крупных приложений так или иначе используют для хранения данных эти системы управления базами данных. Однако чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология ADO.NET.

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Причем важно отметить, что систем управления баз данных может быть множество. В своей сущности они могут различаться. MS SQL Server, например, для создания запросов использует язык T-SQL, а MySQL и Oracle применяют язык PL-SQL. Разные системы баз данных могут иметь разные типы данных. Также могут различаться какие-то другие моменты. Однако функционал ADO.NET построен таким образом, чтобы предоставить разработчикам унифицированный интерфейс для работы с самыми различными СУБД.

Чтобы использовать один и тот же набор объектов для разных источников данных, необходим соответствующий провайдер данных. Собственно через провайдер данных в ADO.NET и осуществляется взаимодействие с базой данных. Причем для каждого источника данных в ADO.NET может быть свой провайдер, который собственно и определяет конкретную реализацию вышеуказанных классов.

По умолчанию в ADO.NET всторен провайдер для MS SQL Server.

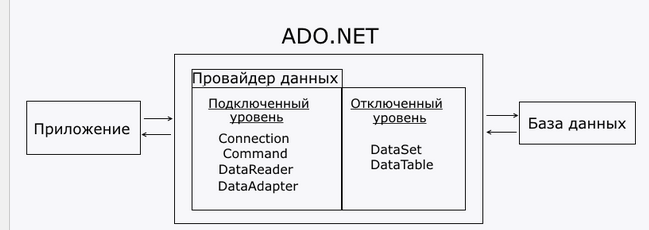
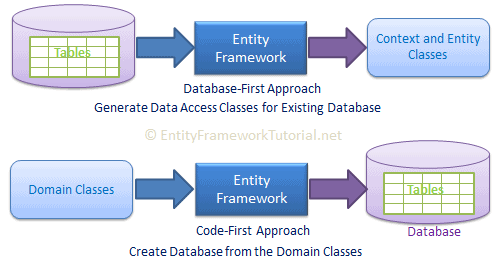


Рисунок 23 – схематичная архитектура ADO .NET



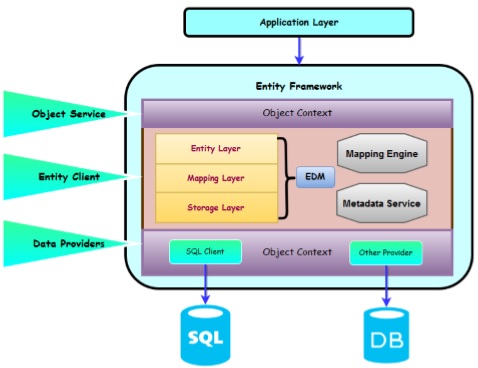


Рисунок 24 – представление технологии передачи данных между компонентами

## Реализация функционирующего приложения

### Серверный компонент

Database First был первым подходом, который появился в Entity Framework. Данный подход во многом похож на Model First и подходит для тех случаев, когда разработчик уже имеет готовую базу данных.

Чтобы Entity Framework мог получить доступ к базе данных, в системе должен быть установлен соответствующий провайдер. Так, Visual Studio уже поддерживает соответствующую инфраструктуру для СУБД MS SQL Server.

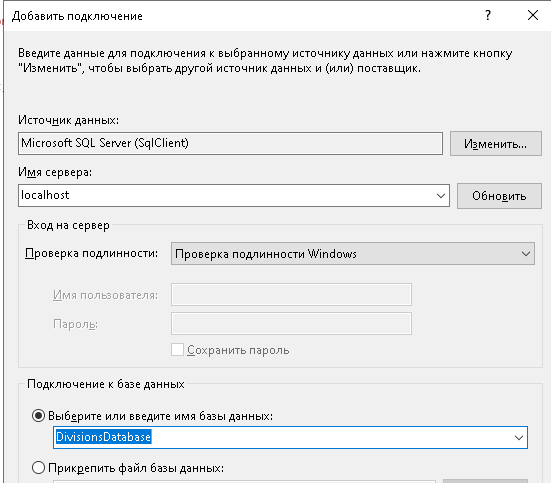


Рисунок 25– подключение к базе данных

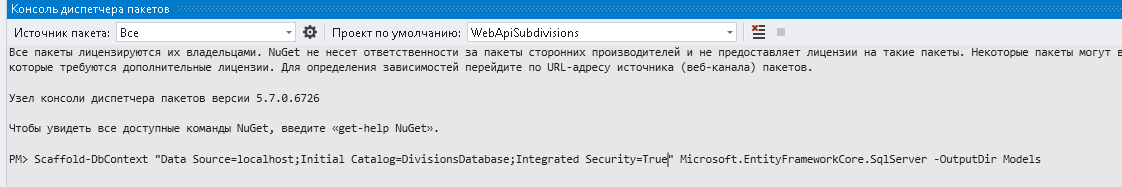


Рисунок 26 – скрипт для генерации DBContext и классов из сущностей для базы данных

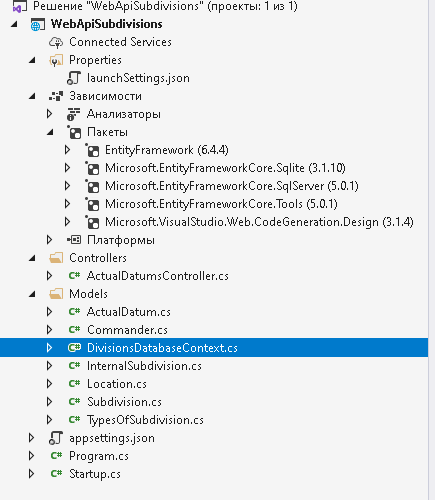


Рисунок 27 – структура проекта

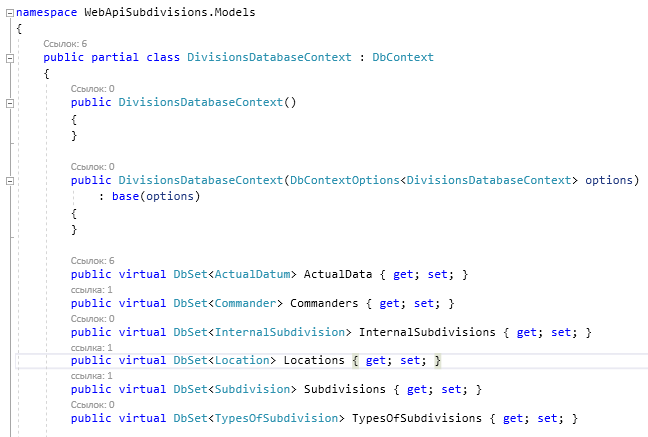


Рисунок 28 – листинг свойств класса DivisionsDatabaseContext



Рисунок 29 – добавление в настройках проекта строки подключения к базе данных

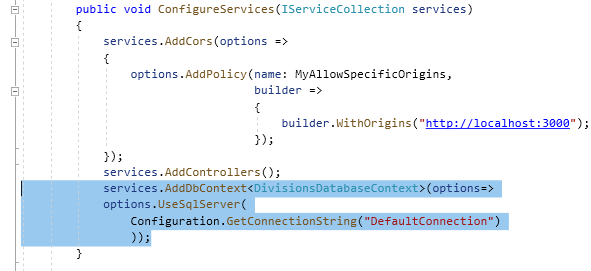


Рисунок 30 – добавление класса контекста в сервисы приложения

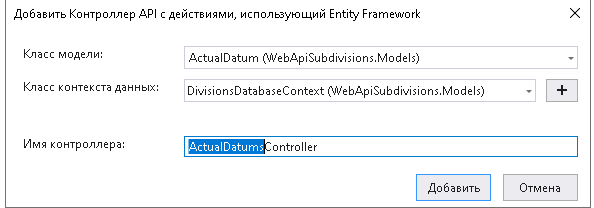


Рисунок 31 – добавление контроллера

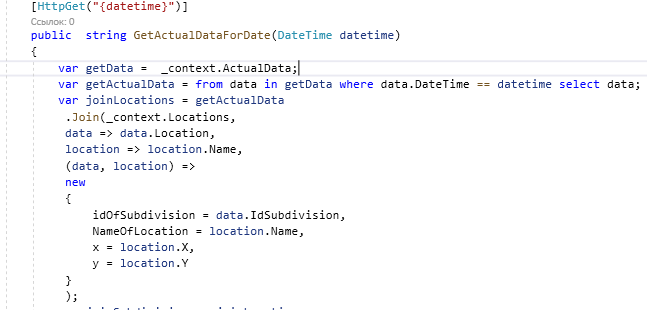


Рисунок 32 – основной метод возвращения актуальной информации

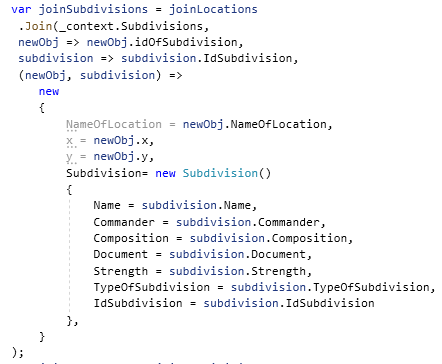


Рисунок 33 – запрос join Subdivisions



Рисунок 34 – запрос join Commanders



Рисунок 35 – обработка данных для передачи в формате JSON

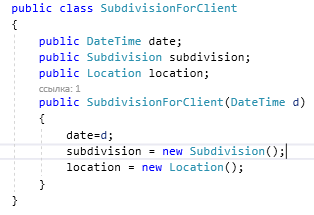


Рисунок 36 – дополнительно созданный класс для отправки клиенту

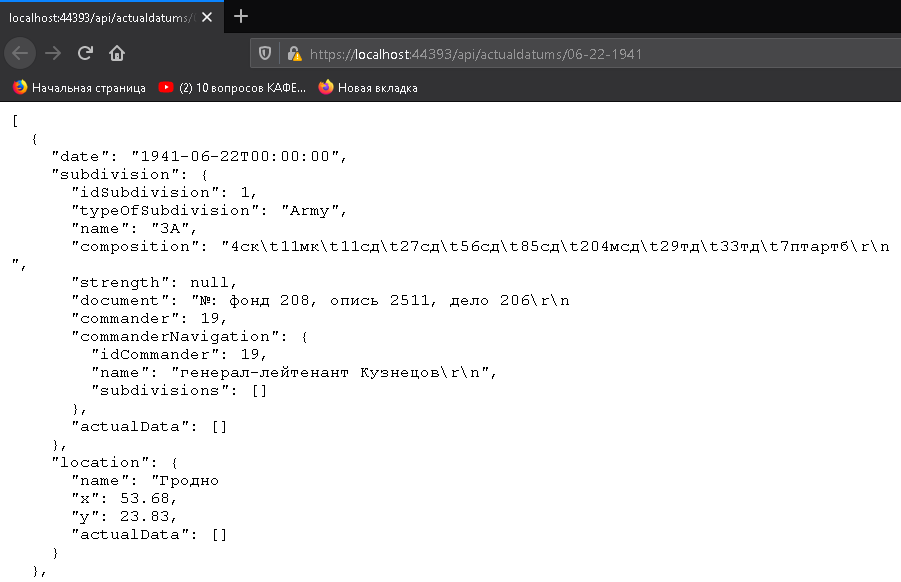


Рисунок 37 – возвращаемый с сервера объект

### Клиентское приложение

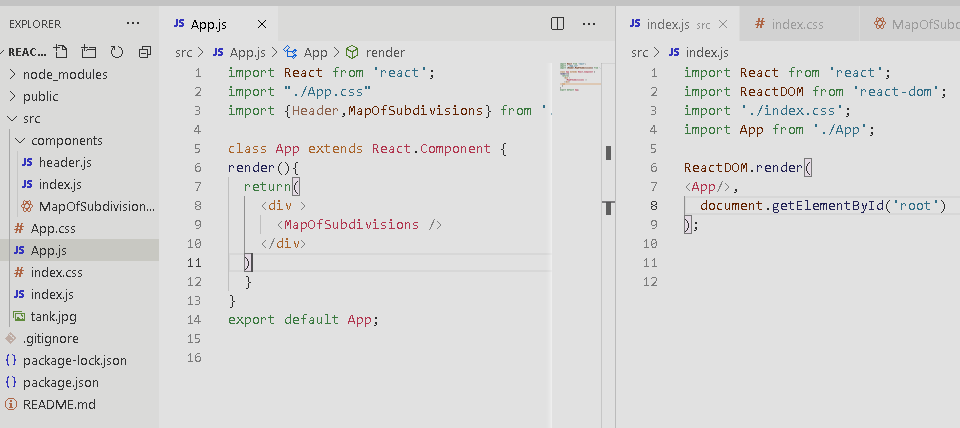


Рисунок 38 – корневые элементы клиентского приложения

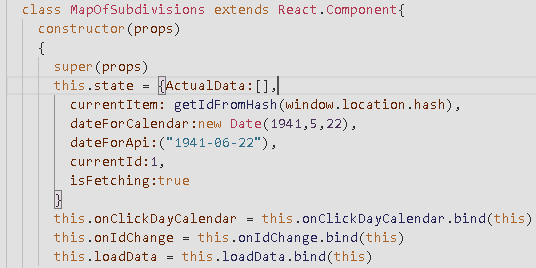


Рисунок 39 – конструктор карты (React компонента)

На стадии жизненного цикла «монтирование компонента» необходимо загрузить данные с сервера и определить состояние приложения.



Рисунок 40 – посылка запроса и установка первоначального состояния приложения

В методе render формируется код отрисовки элементов на странице веб-браузера.

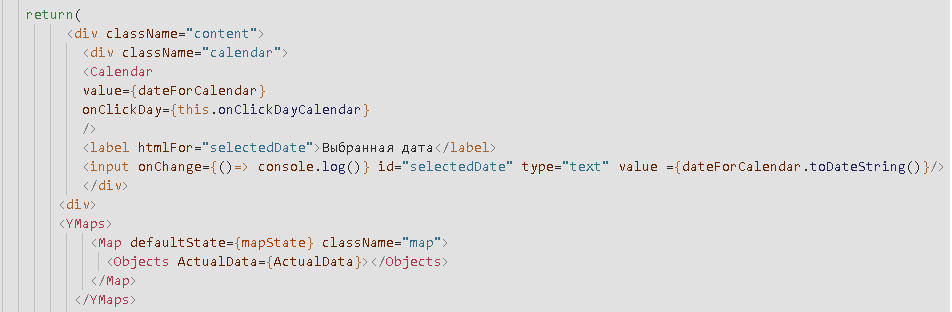


Рисунок 41 – метод render



Рисунок 42 – формирование кода отрисовки элементов



Рисунок 43 –формирование кода отрисовки элементов

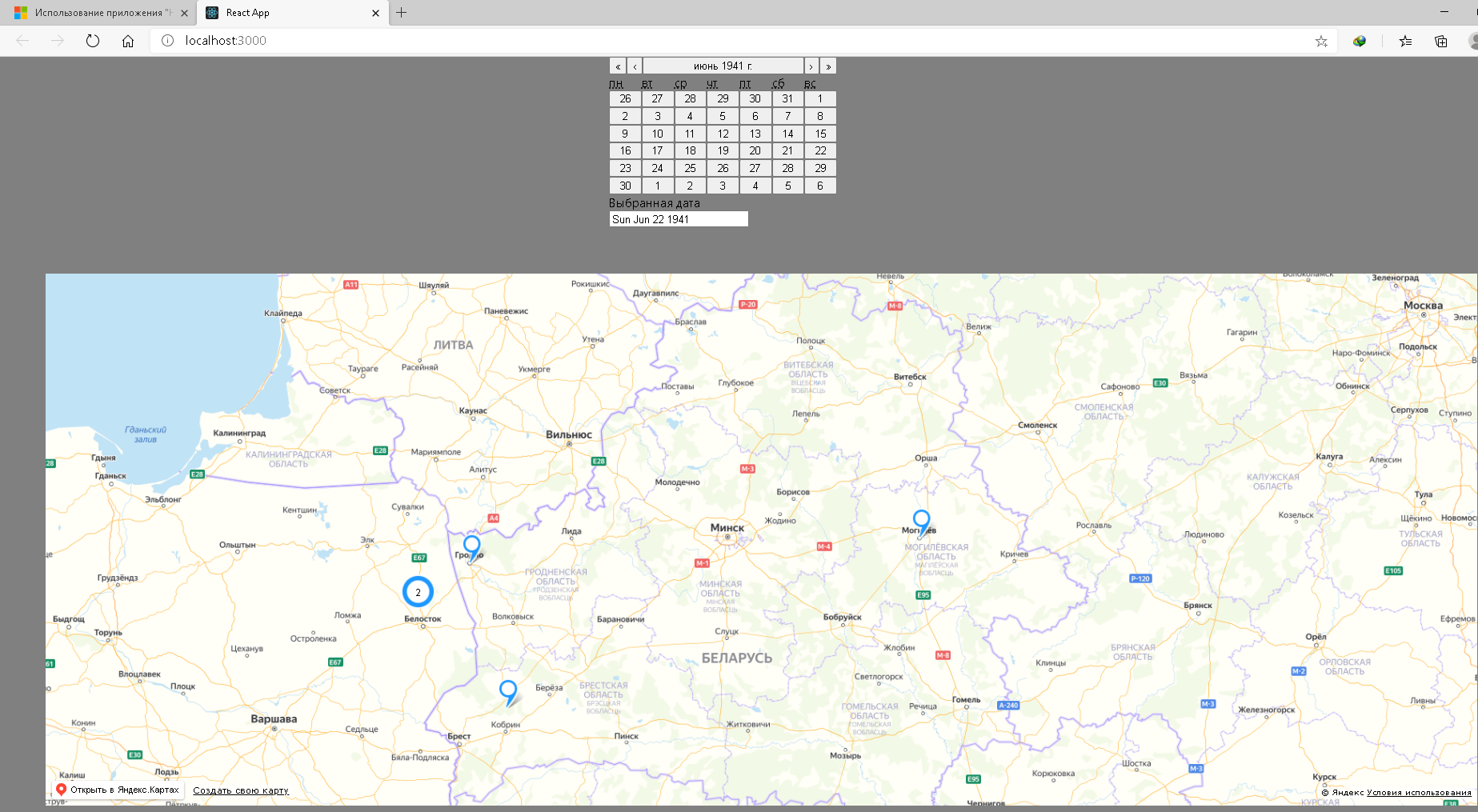


Рисунок 44 – интерфейс приложения



Рисунок 45 – интерфейс приложения

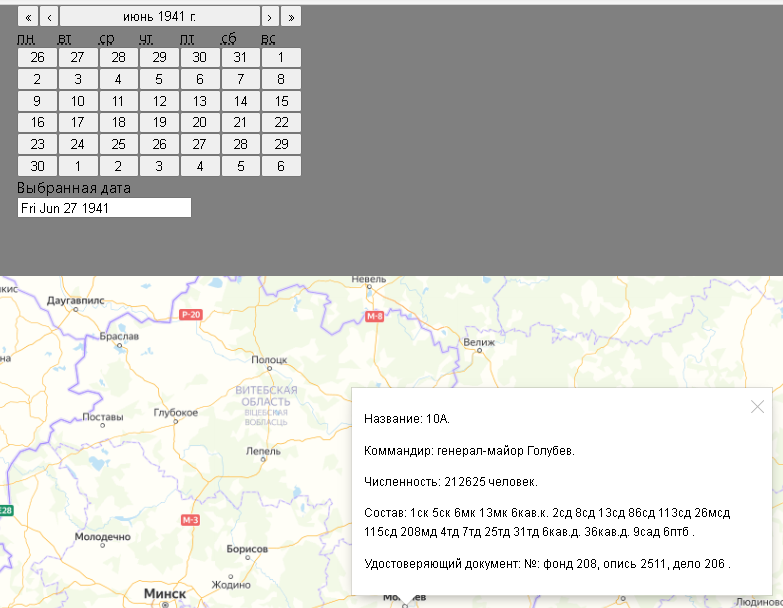


Рисунок 46– отображение информации о подразделении

Заключение: в результате выполнения курсовой работы все задачи были выполнены, а цель достигнута.

# Список литературы

<https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema7/tema7_2>

<http://army.armor.kiev.ua/hist/ierarx.shtml>

Приложение 1 .Концептуальная модель предметной области

