

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Разработка веб-базируемого микросервиса
«Календарь памятных дат Петроградской стороны»

Обучающегося 4 курса
очной формы обучения
Лазебниковой Полины Михайловны

Руководитель выпускной квалификационной
работы:
Кандидат физ.-мат. наук, доцент кафедры
ИТиЭО
Жуков Николай Николаевич

Рецензент:
Ученая степень (*при наличии*), ученое звание
(*при наличии*), должность
Ф. И. О. (*указывается в именительном
падеже*)

Санкт-Петербург
2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
ГЛАВА I. ВЕБ – БАЗИРОВАННЫЕ СЕРВИСЫ, ПОСВЯЩЕННЫЕ ПАМЯТНЫМ СОБЫТИЯМ	5
1.1 Введение в микросервисную архитектуру	5
1.2 Анализ существующих сервисов, посвященных памятным датам	7
1.3 Анализ существующих технологий для реализации данного сервиса	14
1.4 Жизненный цикл программного продукта	19
1.5 Выводы к главе I	26
ГЛАВА II. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ВЕБ-БАЗИРОВАННОГО МИКРОСЕРВИСА «КАЛЕНДАРЬ ПАМЯТНЫХ ДАТ ПЕТРОГРАДСКОЙ СТОРОНЫ».....	27
2.1 Этапы жизненного цикла микросервиса	27
2.2 Разработка микросервиса.....	33
2.3 Выводы к главе II.....	41
ЗАКЛЮЧЕНИЕ	42
ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ	43
ПРИЛОЖЕНИЯ.....	49
ПРИЛОЖЕНИЕ А	49
ПРИЛОЖЕНИЕ Б.....	50

ВВЕДЕНИЕ

Каждый человек является частью какого-либо народа, у которого есть свой язык, культура и история. Именно это делает его единым. Поэтому патриотическое воспитание является одним из основных составляющих формирования личности.

У каждого народа есть государственный календарь, в котором отражены важные памятные даты. Также для области, региона, города или конкретного района может существовать свой собственный календарь исторических событий. Календари содержат наиболее значимые даты исторических событий, профессиональные и всемирные праздники, юбилеи жизни деятелей искусства и культуры, ученых, отечественных писателей и другие важные даты.

Для того, чтобы сделать такой календарь общедоступным, его можно разместить в интернете в качестве веб-ресурса. Аналогичные проекты уже существуют по различным регионам России. Проведя анализ некоторых из них, были выявлены существенные недостатки, которые описаны в первой главе данной работы. У заказчика проекта уже есть подобный сайт, за время использования которым были выявлены различные недостатки, появилась потребность в новых элементах. Текущий сайт сделан с помощью системы управления статическим контентом Lector, поэтому редактирование как его наполнения, так и добавления новых функций затруднительно. Одним из вариантов решения данной проблемы стала разработка нового микросервиса с учетом всех требований, пожеланий, недочетов. В этом и заключается **актуальность** работы, посвященной разработке веб-базируемого микросервиса, посвященного памятным датам Петроградской стороны по заказу Санкт-Петербургского государственного бюджетного учреждения «Централизованная библиотечная система Петроградского района».

Объект исследования: процесс разработки веб-базируемого микросервиса, посвященного знаменательным датам.

Предмет исследования: разрабатываемый веб-базируемый микросервис.

Цель работы: разработка веб-базируемого микросервиса «Календарь памятных дат Петроградской стороны» в соответствии с требованиями заказчика.

Для достижения поставленной цели было необходимо решить следующие **задачи**:

1. Провести анализ существующих решений, посвященных памятным датам и выявить главные отличия разрабатываемого сервиса.
2. Спроектировать архитектуру микросервиса.
3. Спроектировать схему базу данных.
4. Разработать docker – образ микросервиса.
5. Разработать фронтенд и бэкенд на Django.
6. Разработать API для дальнейшей возможности переписи фронтенда сервиса с использованием других инструментов.

Работа включает в себя 50 страниц, 26 рисунков и 2 таблицы.

ГЛАВА I. ВЕБ – БАЗИРОВАННЫЕ СЕРВИСЫ, ПОСВЯЩЕННЫЕ ПАМЯТНЫМ СОБЫТИЯМ

1.1 Введение в микросервисную архитектуру

На сегодняшний день существуют разные типы архитектуры веб-приложений, с помощью которых разработка и управление системой становится проще, и работа системы становится более предсказуемой. Одни из самых известных архитектур [28]:

- Многослойная архитектура (Layered Architecture).
- Микросервисная архитектура (Microservice Architecture).
- Многоуровневая архитектура (Tiered Architecture).
- Сервис-ориентированная архитектура (Service Oriented Architecture — SOA).

Для разрабатываемого проекта наиболее подходит микросервисная архитектура. Ниже рассмотрено, почему именно ее стоит выбрать, а также особенности, преимущества и недостатки данного типа.

Микросервисная архитектура веб-приложений — это концепция, предполагающая реализацию приложения в виде набора небольших сервисов, характеризующихся собственными функциями, параметрами коммуникации, характером исполнения и т.д. Таким образом, веб-базированный микросервис — это составной элемент архитектуры приложения, основанный на веб-технологиях (для обмена данными, в большинстве случаев, используется HTTP [20]). В совокупности микросервисы определяют общую функциональность веб-приложения на глобальном уровне. В зависимости от масштабов проекта, микросервис сам по себе может представлять собой емкое приложение, либо небольшой веб-модуль. С этой точки зрения важно лишь соблюдение концепции — представление приложения в виде автономных

сервисов, «жизнеспособность» каждого из которых не зависит от других. В качестве механизма взаимодействия микросервисов могут использоваться HTTP и REST, JSON или XML.

Являясь альтернативой монолитной архитектуре построения веб-базированного приложения, при которой вся работа приложения организована на одной кодовой базе [19], микросервисная архитектура повышает уровень надежности приложения, поскольку выход из строя или изменения в рамках одного сервиса не повлияют на работу других, в отличие от монолита. Стоит отметить и другие преимущества микросервисов:

- доступно масштабирование на уровне отдельного сервиса;
- понятная и прозрачная структура;
- быстрая работоспособность;
- ускорение обновлений за счет отсутствия сложной структуры связей;
- возможность использования собственного стека технологий на уровне каждого сервиса.

Из недостатков можно выделить:

- трудность в управлении нескольких сервисов;
- на реализацию нескольких сервисов потребуется больше времени;
- высокий риск сбоя при обмене данными между сервисами.

Недостатки связаны с большим количеством сервисов, что не относится к текущей разработке. Исходя из вышеперечисленного можно сделать вывод, что микросервисная архитектура является наиболее подходящей для разработки в рамках выпускной квалификационной работы.

1.2 Анализ существующих сервисов, посвященных памятным датам

В ходе работы были проанализированы следующие решения, посвященные памятным датам:

- Календарь памятных дат Сибирского отделения Российской академии наук (СО РАН) [12];
- Календарь знаменательных дат МАУК «Муниципальной информационно-библиотечная система» г. Кемерово [4];
- Календарь знаменательных и юбилейных дат Московской области [5];
- Календарь знаменательных дат Карелии [8];
- Календарь исторических событий и знаменательных дат г. Ижевска [10];
- Хронологический справочник «Имена на карте Ленинградской области» [26].

На следующем этапе исследования выполним подробный разбор функций, дизайна, эргономических особенностей и прочих критериев представленных сервисов.

1. Календарь памятных дат СО РАН

Формат представления данного календаря имеет подобие с электронной книгой. По аналогии с перелистыванием страниц выбирается год, начиная с 2010 (по горизонтали) и месяц (по вертикали). Календарь имеет очень ограниченное назначение – в нем указаны только даты и события, имеющие значения только в контексте деятельности СО РАН. Поскольку данным календарем, по большей части, пользуются сотрудники СО РАН, большинство из которых – люди в возрасте, в качестве основных недостатков следует отметить мелкий шрифт, используемый для описания. Дополнительно затрудняют навигацию множество дополнительных мелких ссылок, которые не имеют большой значимости конкретно для календаря. В качестве полезной функции стоит отметить наличие ссылок в описании событий, т.е. имеется

возможность сразу перейти к описанию той или иной персоналии, научного труда или мероприятия. Несмотря на привычный формат электронной книги, календарь не совсем удачен с точки зрения юзабилити, однако, он, несомненно, имеет большую пользу и ценность для сотрудников СО РАН. Главная страница данного календаря изображена на рисунке 1.



Рисунок 1 – Главная страница календаря памятных дат СО РАН

2. Календарь МАУК «МИБС» г. Кемерово

Календарь посвящен знаменательным датам г. Кемерово и Кемеровской области. Календарь достаточно удобен в использовании – в отдельной области электронного календаря выбирается месяц и день, затем в основной области для чтения подгружаются имеющиеся в базе данных даты и события. Отдельным образом следует отметить наличие картинок, соответствующих событиям, но при этом отсутствуют ссылки на подробные описания. Поскольку в базе данных календаря достаточно много событий, существенным минусом является отсутствие механизма поиска по датам или событиям. Отсутствует категоризация – все события на конкретный месяц представлены скопом на одной странице (дни рождения людей, праздники, события и т.д.), что существенно затрудняет навигацию по календарю. Главная страница календаря изображена на рисунке 2.

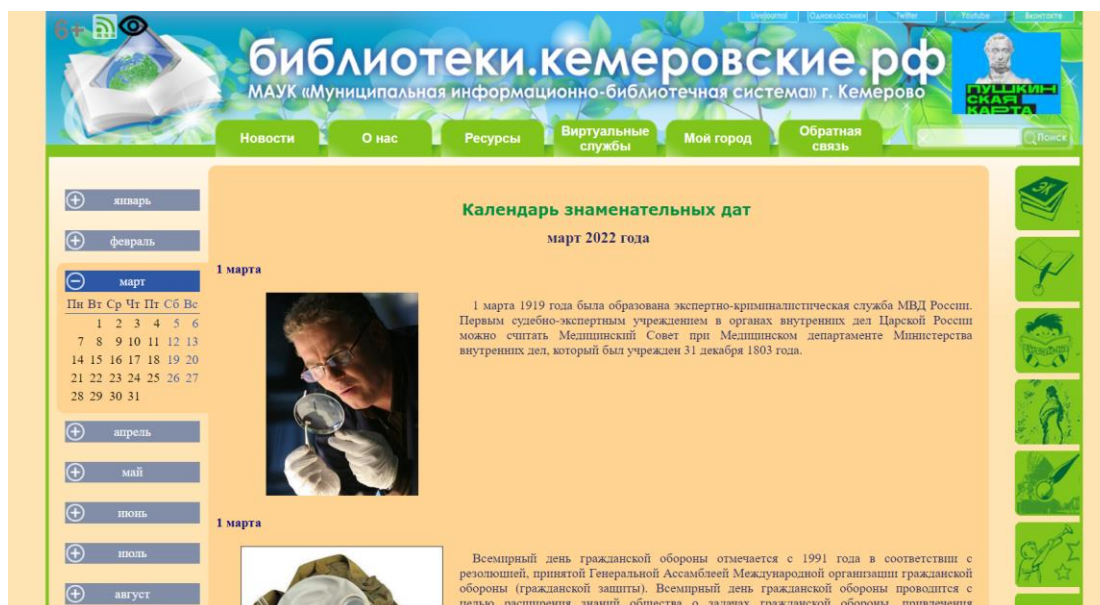


Рисунок 2 – Главная страница календаря МАУК г. Кемерово

3. Календарь знаменательных и юбилейных дат МО

Значимые события для Москвы и МО представлены в виде интерактивного электронного календаря, показанного на рисунке 3, который занимает практически всю область сайта. События в календаре выстроены в виде сетки с днем недели и числом. Такой подход к представлению, с одной стороны, удобен – можно быстро найти интересующее событие в пределах какого-то месяца. При нажатии на определенную дату выводится дополнительная табличка и более крупным шрифтом показываются вложенные события, если они есть.

Из минусов отмечены следующие: отсутствие подробного описания событий (доступны только дата и наименование события), а также отсутствие механизма поиска. Второй минус очень существенен – из-за отсутствия быстрого поиска даже на уровне года придется потратить много времени, чтобы добраться, например, до 1960-го года (нужно пролистать вручную более 60-ти лет назад). Кроме этого, последнее событие датируется декабрем прошлого года – начиная с нынешнего года учет дат не ведется, что говорит, скорее всего, об отсутствии заинтересованности поддержки данных сервиса в актуальном состоянии со стороны администраторов.

Предположительно после добавления механизмов поиска и подробного описания событий в интерактивной выпадающей табличке, настоящий сервис мог бы претендовать на высокий уровень.

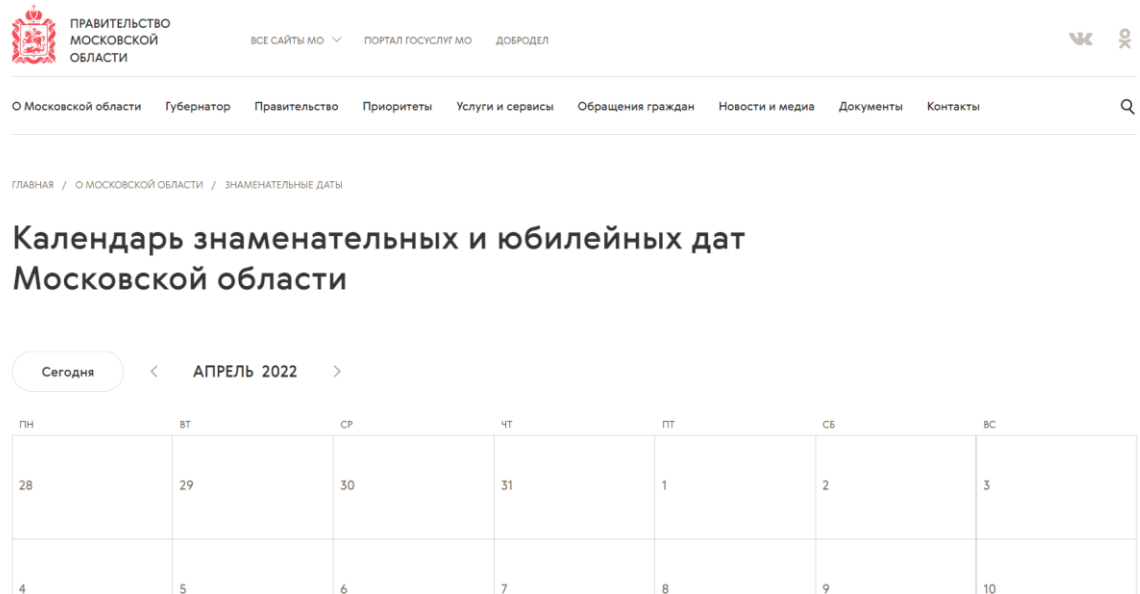


Рисунок 3 – Главная страница календаря знаменательных и юбилейных дат МО

4. Календарь знаменательных дат Карелии

Для данного календаря стоит отметить маленький временной охват – знаменательные даты представлены только на 2021 год. В пределах этого года в интерактивной области выбрать можно месяц, число и день, после чего пользователь в табличном представлении (колонки дата и событие) получит список дат и событий.

С точки зрения наполнения следует отметить полезные ряд положительных моментов, например, автоматический расчет прошедшего времени в года со дня события, а также указание библиографической ссылки на литературу, которая связана с тем или иным памятным событием республики. Имеется категоризация по персонам, географическим названиям и событиям. Предусмотрен поиск по буквам алфавита.

Помимо малого временного охвата в качестве основного недостатка следует отметить отсутствие механизма поиска. Это обуславливает проблему

навигации по большому количеству событий на каждый из месяцев, представленных в виде таблицы. Главная страница сайта изображена на рисунке 4.



Рисунок 4 – Главная страница календаря знаменательных дат Карелии

5. Календарь исторических событий и знаменательных дат г. Ижевска

Данный сервис календаря характеризуется слабой адаптацией под пользователя с точки зрения юзабилити при относительно неплохом цветовом решении. Отсутствует описание календаря, пользователю сразу представлен для выбора список месяцев. При выборе месяца в отдельном окне скопом появляются все имеющиеся в базе данных события. Нет хронологической последовательности с точки зрения представления дат (например, после 2017 года может идти 1923). В некоторых местах наблюдается разный шрифт. Отсутствуют даже примитивные механизмы навигации и поиска, что значительно затрудняет пользование календарем. Из интерактивных функций – только выбор месяца и всплывающее окно. Данный календарь, показанный на рисунке 5, можно считать самым неудачным решением из представленных сервисов, посвященных памятным датам. Он практически непригоден для прикладного использования.

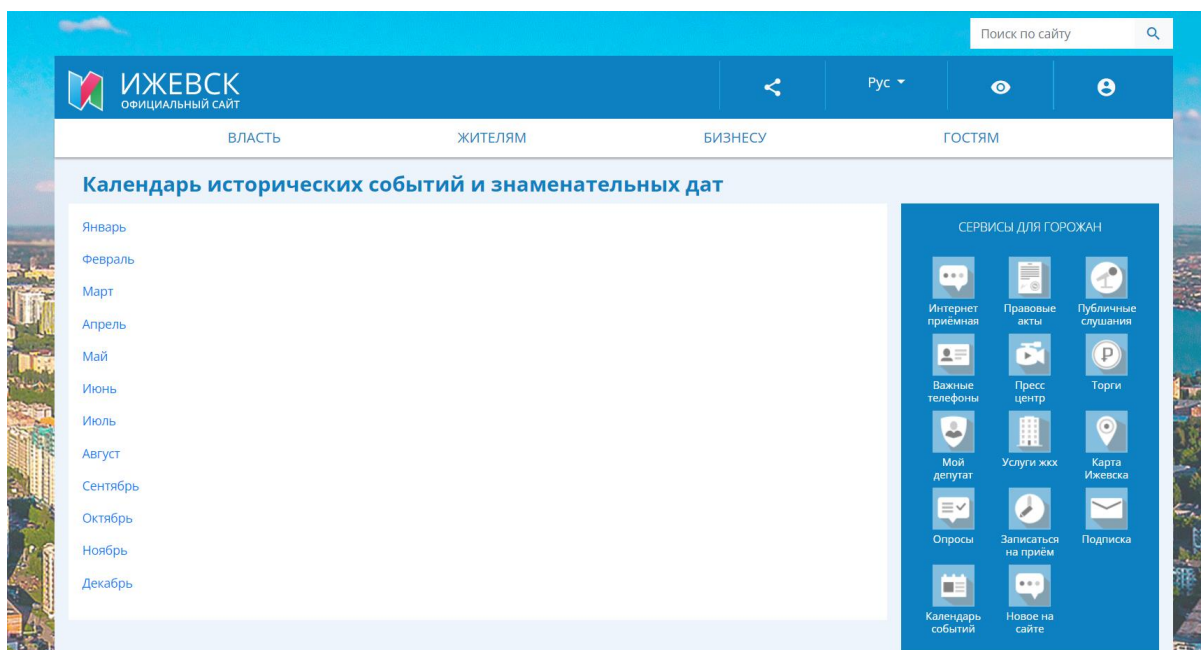


Рисунок 5 – Главная страница календаря исторических событий и знаменательных дат г. Ижевска.

6. Хронологический справочник «Имена на карте Ленинградской области»

Отдельным образом стоит рассмотреть сервис Ленинградской научной библиотеки, изображенный на рисунке 6. Он посвящен значимым людям, местам и родам деятельности Ленинградской области и Петербурга. В рамках использования сервиса предусмотрены следующие категории:

- персоналии;
- места;
- роды деятельности.

Глобально сервис посвящен известным личностям. В качестве полезной функции следует отметить возможности расширенного поиска персоналий по ряду критериев: имя, место, дата рождения, род деятельности. При начале ввода система подсказывает возможные варианты, содержащиеся в базе, например, для населенных пунктов и видов деятельности. Каждая страница содержит подробное описание личности с расчетом прошедших лет со дня

рождения и с даты смерти. В каждую страницу с описанием личности интегрирована электронная карта с отметкой места рождения.

Из недостатков сервиса следует отметить возможность поиска людей по событиям, с которыми они связаны, а также нюансы оформления – местами друг на друга наезжают текстовые конструкции.

В рамках аналитического обзора были рассмотрены шесть существующих решений микросервисов, посвященных памятным датам и личностям. Каждый из них характеризуется собственной спецификой, преимуществами и недостатками. Среди основных недостатков рассмотренных сервисов по результатам обзора отмечены следующие:

- отсутствие механизмов поиска;
- слабый уровень категоризации;
- относительно малый уровень интерактивности;
- неудачная эргономика (юзабилити) для некоторых случаев.

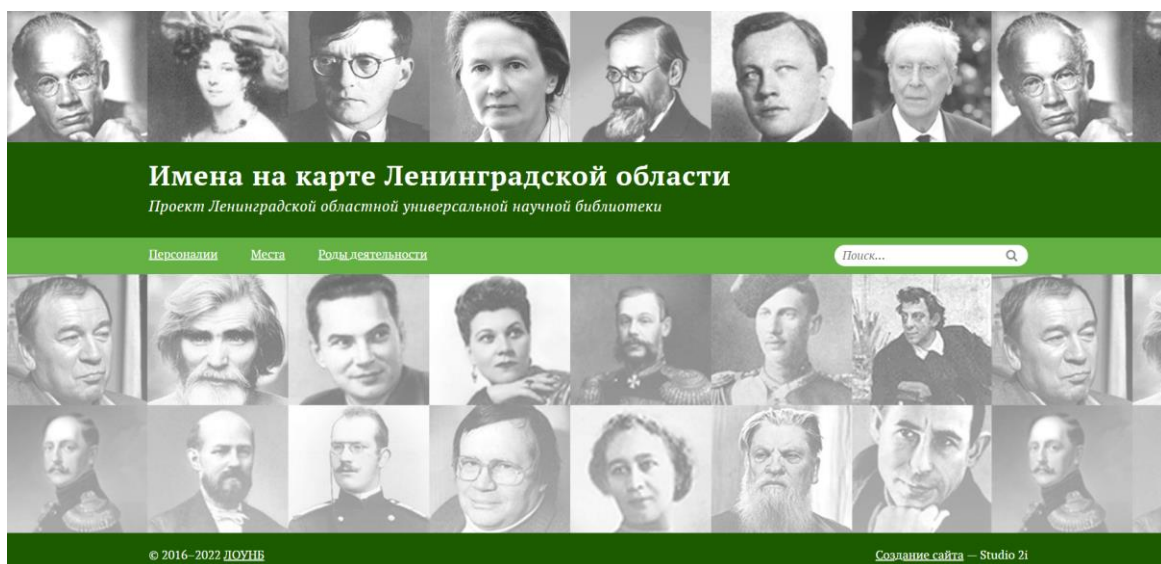


Рисунок 6 – Главная страница хронологического справочника «Имена на карте Ленинградской области»

Разрабатываемый веб-базируемый микросервис памятных дат будет учитывать описанные недостатки. Здесь предусмотрена категоризация по событиям и персонам, внедрен механизм оперативного поиска даты, быстрый переход к предыдущей и следующей дате и другие функции. На главной

странице сервиса представлен список событий с изображениями, что повышает интерес к сервису. Каждое событие характеризуется подробным описанием на собственной странице, что соответствует принципу электронной библиотеки.

1.3 Анализ существующих технологий для реализации данного сервиса

В настоящее время использование программного обеспечения является необходимым условием развития любого направления производства. С целью определения структуры программной системы используется программная платформа, получившая название фреймворк.

Фреймворк представляет собой программное обеспечение, которое позволяет облегчить разработку программного проекта. Его основным назначением является объединение различных компонентов [17]. Очень часто при термине «фреймворк» используется «каркасный подход», т.е. это подход к построению программы, который основан на 2 частях:

- не изменяющая конфигурация, «скелет» программы или постоянная часть, в которой размещаются переменные части;
- сменные модули, которые позволяют выполнять различные функции.

Понятие фреймворка близко к понятию программной библиотеки, но их различие заключается в том, что библиотека может использоваться в программном продукте в качестве набора подпрограмм близкой функциональности. При этом влияние на архитектуру программного продукта не оказывается. Фреймворк же, в свою очередь, осуществляет руководство правилами построения архитектуры приложения. Изначально задаётся каркас, следом он дополняется на основе требований, предъявляемых к программному продукту.

Существуют различные типы фреймворков, в данной работе были рассмотрены следующие четыре: Django, Flask, ExpressJS, Laravel. Проведем

анализ представленных фреймворков, оценку их достоинств, недостатков и области использования.

Django

Django считается один из ведущих фреймворком. Его основу составляет доступный язык программирования Python. Данный фреймворк может быть использован как для разработки сложных и многофункциональных веб-сайтов на основе баз данных, так и для более простых. Одной из отличительных особенностей Django является наличие интерфейса администратора, помогающего создавать и модернизировать различные операции [21]. Среди преимуществ Django стоит выделить его простоту использования, позволяющую ускорить процесс разработки. Кроме того, имеется целый ряд функций, которые помогают при создании карт сайта, администрировании контента и т.д.

Большинство крупных компаний привыкли решать задачи в одном приложении, поэтому важным свойством для них является масштабируемость. В случае с Django проблем с реализацией высоких требований в области эксплуатационных потребностей нет. В целом, в Django имеются различные встроенные инструменты, позволяющие решить возникающие трудности «здесь и сейчас». Django имеет собственную систему создания для всех инструментов и функций. Кроме того, он также имеет простую в использовании панель администратора по сравнению с Flask и Lavarel [29].

Основные преимущества Django:

- Принцип «Все включено».

Большинство инструментов для разработки являются частью фреймворка.

- Стандартизированная структура.

Django задает структуру проекта.

- REST Framework для создания API, имеющий настраиваемую и модульную архитектуру.

Flask

Flask представляет собой небольшой веб-фреймворк, основу которого также составляет язык Python. При этом нет необходимости в использовании дополнительных библиотек или инструментах. Ещё одна особенность Flask в том, что он может поддерживать расширения, которые были добавлены в роли функции в стороннем приложении. Минус Flask в том, что он немного уровней представления между пользователем, запросами и базой данных. Поэтому его обычно применяют для ряда простых задач [3].

ExpressJS

ExpressJS включает в себя открытый исходный код, который доступен по лицензии. Область применения ExpressJS — веб-приложения, API-интерфейсы. При этом сразу же виден недостаток ExpressJS, который заключается в том, что данный фреймворк предназначен для стандартной серверной среды Node.js. То есть его использование обычно происходит совместно с интерфейсной структурой AngularJS и базами данных.

В качестве языка программирования используется JavaScript, поэтому нет необходимости в больших временных затратах для освоения данного фреймворка. В целом, все особенности данного фреймворка завязаны на Node.js. К примеру, с его помощью разработчики могут применять Java для создания серверных приложений или клиентских приложений. Таким образом, нет нужды в подключении фронтенд и бэкенд разработчиков. Также стоит отметить весьма удобную систему отладки в указанном фреймворке.

Laravel

Laravel использует модульную систему вместе со специальным менеджером зависимостей. Данный фреймворк позволяет пользователям иметь несколько вариантов доступа к базам данных или утилитам для обслуживания. Преимуществами Laravel является лёгкая настройка и упрощённая логика авторизации. У Laravel имеется целый ряд шаблонов, позволяющих упростить процесс разработки [13]. Шаблоны включают в себя

макеты с различными механизмами, позволяющими разделять уровни представления и бизнес-логику. Кроме того, имеется целый ряд виджетов.

Среди недостатков Laravel стоит выделить ограниченную поддержку по причине своей легкости. Поэтому приходится использовать сторонние средства для ухода от данной проблемы. Например, при работе изначально с Laravel вы не увидите дебаг панель [11], что создаёт затруднения при отладке. Кроме того, у Laravel нет базовых классов, при этом модели в процессе генерации получают от модели из вендора. Следовательно, создаются проблемы с расширением.

В таблице 1 представлена сравнительная характеристика выделенных фреймворков, содержащая:

- основной язык программирования;
- основные особенности рассмотренных фреймворков;
- наиболее популярные варианты их использования.

Таблица 1 – Сравнительная таблица рассмотренных фреймворков

Фреймворк	Язык программирования	Основные особенности	Наиболее популярные варианты применения
Django	Python	Быстрая структура. Многофункциональность. Безопасность. Масштабируемость. Открытый исходный код.	Instagram Mozilla Coursera
Flask	Python	Гибкость. Простота разработки. Модульный характер.	Red Hat Rackspace Reddit
ExpressJS	NodeJS	Единый язык программирования.	MySpace GeekList

		Простота обучения. Маршрутизация.	Storify
Laravel	PHP	Аутентификация. Журналы. Механизм шаблонов. Безопасность.	Deltanet Travel Neighborhood Lender MyRank

Не существует одного идеального решения, подходящего для каждого проекта, так как все они отличаются. Выбор средства для реализации зависит от многих факторов, задач и требований. Поэтому выбирать фреймворк стоит конкретно под проект, предварительно рассмотрев особенности разработки, преимущества и недостатки.

Рассмотрев различные фреймворки, их особенности, преимущества и недостатки, для разработки в рамках выпускной квалификационной работы был выбран Django. Он задает определенную структуру проекта, включает в себя большинство инструментов для разработки, позволяет создавать REST API. Также данный инструмент является знакомым, поэтому не требуется изучение с нуля. Его удобно использовать для сервисов, работающих с базами данных. В фреймворке все время появляются новые возможности, а также у него большое сообщество, благодаря которому не возникнут большие трудности, так как специалисты уже решали многие проблемы и делились решением в интернете.

1.4 Жизненный цикл программного продукта

Любой проект должен быть определен и спланирован, также он должен обеспечиваться процессами управления рисками, качеством, объемом. Процесс успешной реализации проекта непосредственно зависит от правильно подобранной модели управления.

Моделью жизненного цикла называется совокупность процессов действий, осуществляемых при разработке, внедрении и сопровождении программного продукта.

На любом этапе происходит определенная последовательность процессов, каждый из которых способствует образованию определенного продукта, используя необходимые ресурсы. Процессы делятся на набор действий, а действия, в свою очередь, на ряд задач.

Жизненный цикл проекта представляет собой модель реализации проекта, от степени соответствия которой подходам, используемым для управления, в значительной степени зависит успешность проекта.

Процесс разработки программных проектов должен обеспечить путь от осознания потребностей заказчика до передачи ему готового продукта. Можно выделить следующие этапы данного процесса:

- определение требований;
- проектирование, определение детального состава модулей каждого компонента;
- реализация;
- тестирование;
- эксплуатация и сопровождение готовой системы [5].

Использование таких моделей позволяет производить программное обеспечение высокого качества, которое отвечает всем требованиям заказчиков, позволяет контролировать разработку в условиях ограничений по времени и затратам.

Проанализировав источники [2, 7, 16], были выделены следующие популярные модели жизненного цикла:

- Каскадная модель (Waterfall model).
- Инкрементная модель (Incremental model).
- V – образная модель (V – model).
- Спиральная модель (Spiral model).

Каскадная модель предполагает четко последовательное, единоразовое исполнение этапов по жестким предварительно спланированным требованиям.

Этапы данной модели представлены на рисунке 7.

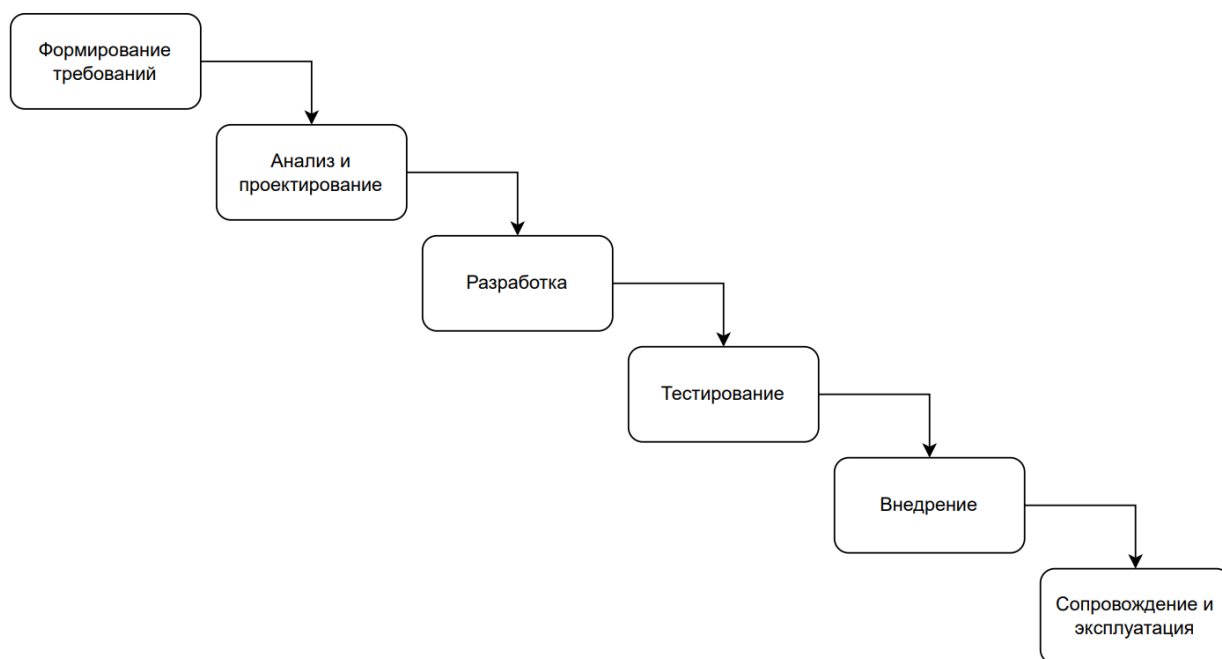


Рисунок 7 – Этапы каскадной модели жизненного цикла

Новый этап начинать нельзя, пока предыдущий не получил статус «завершен». Данная модель может подойти для относительно небольших проектов, цели и требования которых очень четко установлены. Их изменение может повлечь за собой проблемы при реализации проекта.

Каждый этап должен быть заранее спланирован и расположен в нужной последовательности. Изменения требований или приоритетов клиента

приводят к нарушениям последовательности задач, сильно затрудняя управление ими [17].

Каскадная модель подойдет в случае:

- наличия предсказуемого и продуманного плана,
- разработки новой версии существующего решения,
- наличия конкретных установленных требований,
- несложных коротких проектов.

Инкрементная модель предназначена для случаев, когда требуется поэтапная разработка продукта. Ее основой может являться разделение по модулям с ограниченными функциями, что позволит последовательно расширять функционал продукта. Каждый линейный этап подобен каскадной модели [7].

Инкрементная модель представлена на рисунке 8.

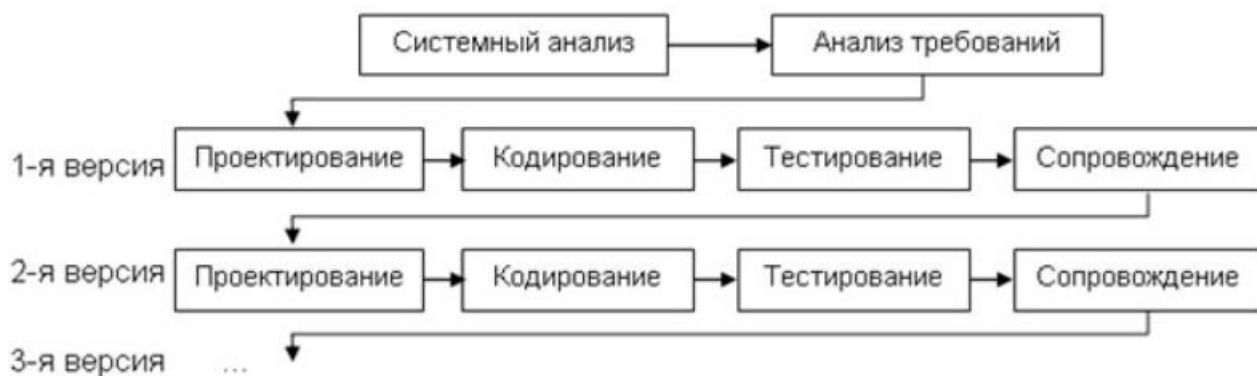


Рисунок 8 – Этапы инкрементной модель жизненного цикла

Разработка ведется инкрементно до самого конца. После первых инкрементов появляется прототип продукта, в котором реализованы главные функции. После каждого следующего инкремента функционал дорабатывается и дополняется.

Инкрементная модель может использоваться при:

- наличии четких требований, которые внедряются пофазово,
- потребности результата на ранних стадиях,
- отсутствии возможности финансирования всего проекта сразу,

- отсутствии нужных ресурсов в начале реализации проекта.

V – образная модель имеет последовательную структуру, как и каскадная модель. Различие заключается в том, что в данной модели каждый этап разработки имеет связанный этап тестирования. По нисходящей ветке отражается процесс разработки, а по восходящей – процесс тестирования.

V – образная модель представлена на рисунке 9.

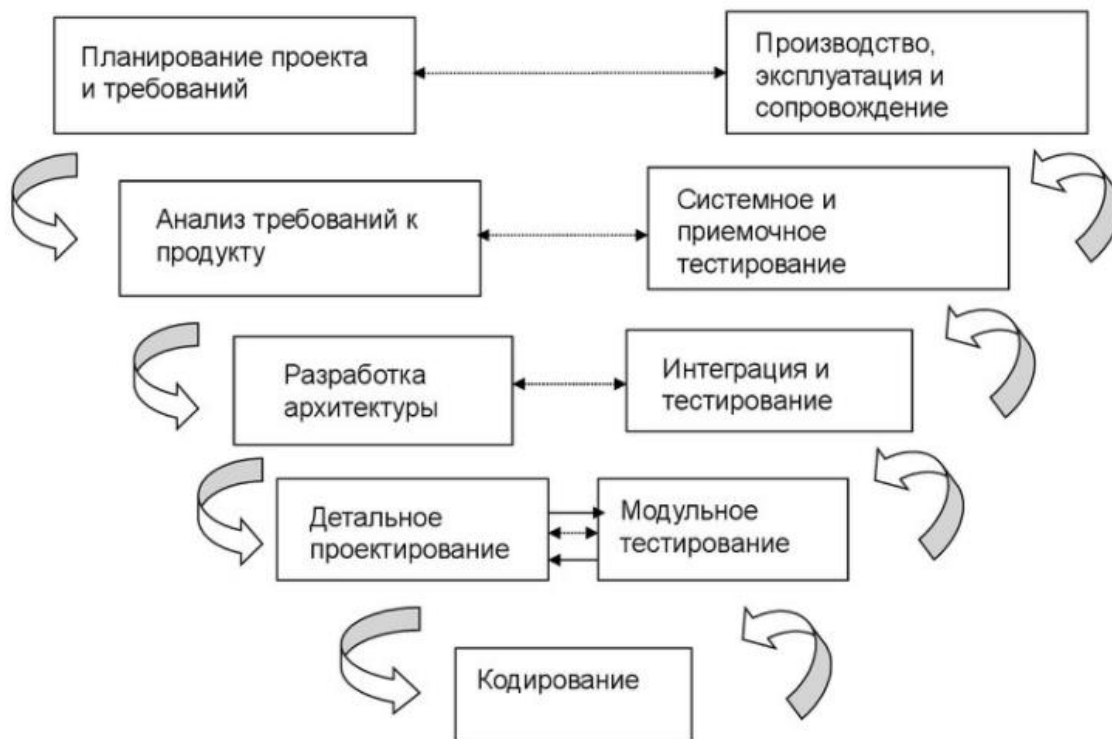


Рисунок 9 – Этапы V – образной модели жизненного цикла

На этапе модульного тестирования происходит тестирование небольших блоков. Для проверки работоспособности нескольких модулей предназначено интеграционное тестирование, а для целой системы – системное. Приемочное тестирование определяет, насколько продукт соответствует всем требованиям, и тестирует его в пользовательской среде.

V – образная модель может использоваться при:

- требовании высокого уровня надежности,
- разработке, для которой требования сформулированы заранее и очень четко,

- наличии большого количества сложных требований,
- разработке, в которой ожидаются изменения или дополнения по требованиям,
- высоком уровне риска,
- необходимости оценки рисков,
- разработке нового продукта.

В таблице 2 рассмотрены достоинства и недостатки выше описанных моделей.

Таблица 2 – Достоинства и недостатки моделей жизненного цикла

Модель	Достоинства	Недостатки
Каскадная	<p>Простота использования.</p> <p>Четкая структура.</p> <p>Стадии выполняются в строго заданном порядке.</p> <p>Требования не изменяются в течение всего цикла.</p>	<p>Нельзя вернуться к предыдущим этапам.</p> <p>Получение результата возможно лишь после прохождения всех этапов.</p> <p>Повышенный риск.</p> <p>Не подходит для долгосрочных сложных проектов.</p> <p>Нет возможности внесения изменений.</p> <p>Недостаточная гибкость.</p>
Инкрементная	<p>Время реализации – более быстрый выход проекта на рынок.</p> <p>Сниженные риски реализации.</p>	<p>В начале реализации необходимо заранее знать количество и функционал каждого</p>

	<p>Функциональный продукт – результат реализации каждого инкремента (модуля).</p> <p>Результат первых итераций является прототипом всей системы.</p> <p>Упрощенный процесс тестирования и внесения правок.</p>	<p>элемента.</p> <p>Постоянное измерение прогресса.</p> <p>Постоянные изменения могут нарушить структуру системы.</p> <p>Дорогостоящая разработка.</p>
V – образная	<p>Постоянное тестирование всех данных на каждом этапе.</p> <p>Упрощение управления и контроля разработки, в отличие от каскадной модели.</p> <p>Надежность и безопасность.</p> <p>Упрощенное использование.</p>	<p>Отсутствие возможности внесения динамических изменений при разработке.</p> <p>Отсутствие анализа рисков.</p> <p>Этапы не могут идти параллельно.</p> <p>Между фазами не предусмотрены итерации.</p>
Спиральная	<p>На поздних этапах могут быть добавлены дополнительные функции.</p> <p>Предотвращение ошибок – постоянный мониторинг рисков и их анализ.</p> <p>Подходит для сложных</p>	<p>Проект может быть довольно затратным.</p> <p>Из-за внесения новых изменений может увеличиваться время разработки проекта.</p> <p>Модель может разбивать</p>

	проектов. Удобство для новых проектов. Гибкое проектирование. Работоспособный продукт может быть выпущен на начальных стадиях разработки.	проект на большое количество промежуточных стадий. Менее применима для небольших проектов.
--	--	--

Разрабатываемый микросервис является сложным проектом, изначально отсутствовало четкое техническое задание, ожидаются изменения и дополнения по ходу работы, поэтому наиболее подходящей моделью жизненного цикла является спиральная модель.

1.5 Выводы к главе I

В главе рассмотрена микросервисная архитектура и обусловлена целесообразность ее использования для разработки данного проекта. Были проанализированы шесть различных сервисов, посвященных памятным датам. Выявлены их сильные и слабые стороны. На основе анализа инструментов для реализации данной работы был выбран фреймворк Django.

Проанализировав четыре популярные модели жизненного цикла, было решено использовать спиральную модель, которая является наиболее подходящей для текущей разработки.

Во второй главе выпускной квалификационной работы описан процесс проектирования и разработки микросервиса с учетом требований заказчика.

ГЛАВА II. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ВЕБ-БАЗИРОВАННОГО МИКРОСЕРВИСА «КАЛЕНДАРЬ ПАМЯТНЫХ ДАТ ПЕТРОГРАДСКОЙ СТОРОНЫ»

2.1 Этапы жизненного цикла микросервиса

Для разрабатываемого проекта была выбрана спиральная модель жизненного цикла. Во время разработки было создано несколько прототипов микросервиса.

Первый прототип

1. Определение требований, проектирование схемы базы данных, макета, выбор инструментов для разработки.

Перед началом работы необходимо было уточнить все требования заказчика. После проведения опроса, вся собранная информация была занесена на доску в Trello. Ее скриншот изображен на рисунке 11. Данный инструмент является популярным сервисом для управления проектами в режиме онлайн. Он имеет простой интерфейс, бесплатный доступ, а также удобен в использовании.

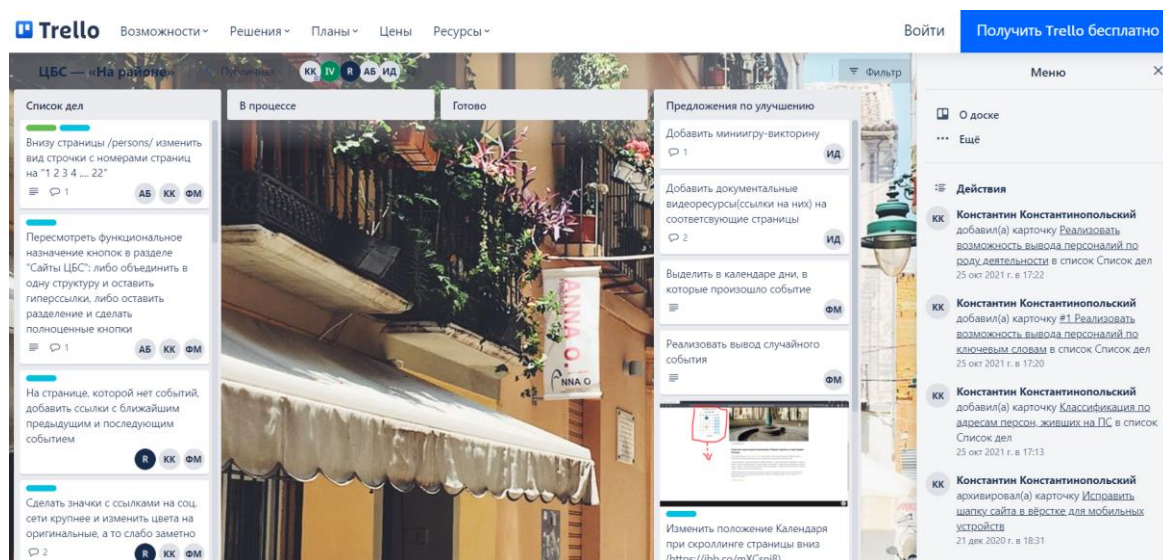


Рисунок 11 – Скриншот доски Trello

Одной из причин решения разрабатывать именно микросервис стали его преимущества, описанные в главе 1.1 данной работы. А другой то, что данный проект является лишь одной из частей общей системы библиотеки, соответственно для дальнейшей работы и связывания ее с другими частями подходила реализация микросервиса, чтобы в дальнейшем все части работали независимо друг от друга, но при этом являлись единым целым.

Разрабатываемый продукт представляет собой веб-базированный микросервис, который разделен на две части: события и персоны. В первом разделе размещены памятные события, с помощью календаря можно выбирать события по датам. Во втором находятся известные люди и деятели Петроградской стороны. Каждое событие и личность сопровождаются краткой справкой.

Требования для раздела «События»:

- добавить теги и улицы для событий,
- для каждого события добавить карусель с книгами,
- ограничить количество выводимых событий на странице,
- изменить положение календаря при скроллинге,
- добавить галерею с событиями месяца в те дни, для которых отсутствуют события.

Требования для раздела «Персоны»:

- добавить теги, улицы, род деятельности для персон,
- для каждой персоны добавить карусель с книгами,
- ограничить количество выводимых событий на странице.

Все функции системы отражены на диаграмме прецедентов в приложении А. Это тип UML (Unified Modeling Language) диаграммы, который показывает типы ролей и их взаимодействие с системой.

В данном проекте следующие роли:

- администратор,
- пользователь.

По описанным требованиям была составлена схема базы данных, представленная на рисунке 13. Она состоит из следующих 9 сущностей:

- Person – Персоны;
- Event – События;
- Person_keyword – Теги персон;
- Event_keyword – Теги событий;
- Profession – Профессии;
- Person_book – Персона – Книга;
- Event_book – Событие – Книга;
- Person_event – Персона – Событие;
- Street – Улицы.

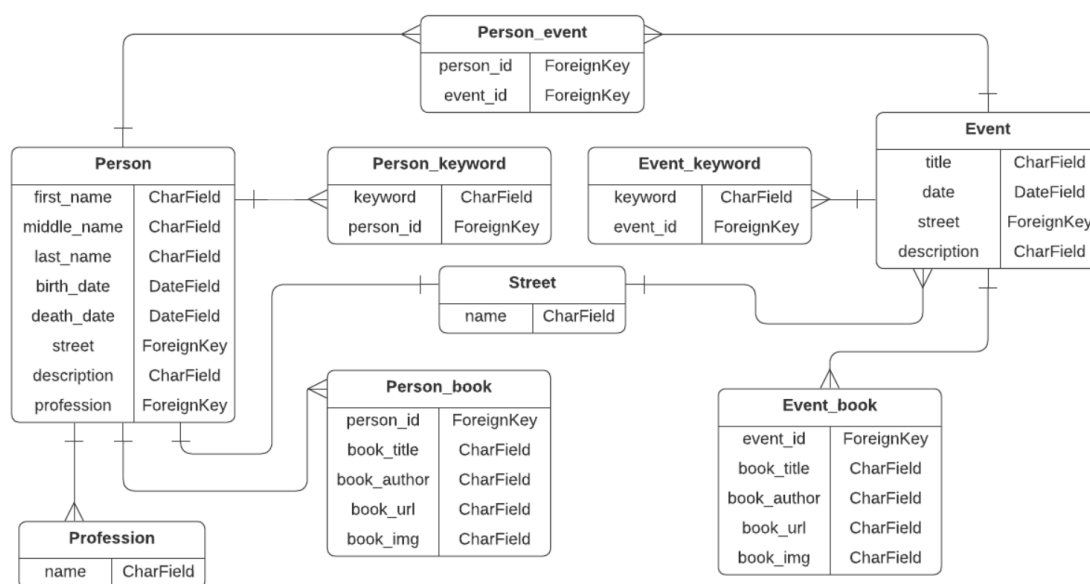


Рисунок 13 – Первый вариант схемы базы данных

2. Создание макета.

Следующий шаг работы – создание макета интерфейса системы. В качестве основы для макета был взят существующий сайт.

Текущий макет содержал недочеты, поэтому было принято решение его изменить. В новом варианте было необходимо:

- сделать дизайн более современным;

- сохранить фирменные отличия библиотеки: бордовый цвет и шрифт Montserrat;
- добавить требуемые элементы;
- сделать интерфейс интуитивно-понятным и удобным для пользователя.

Макеты страниц, созданные с помощью сервиса draw.io, «События» и «Персоны» расположены в приложении Б.

3. Разработка минимума.

В ходе разработки были выявлены проблемы спроектированной базы данных. Необходимо было избежать избыточности и дублирования данных, которые возникали в первом варианте, поэтому было принято решение добавить три новые сущности:

- Person_profession – Профессии персон;
- Keyword – Теги;
- Book – Книги.

Конечный вариант схемы базы данных представлен на рисунке 14.

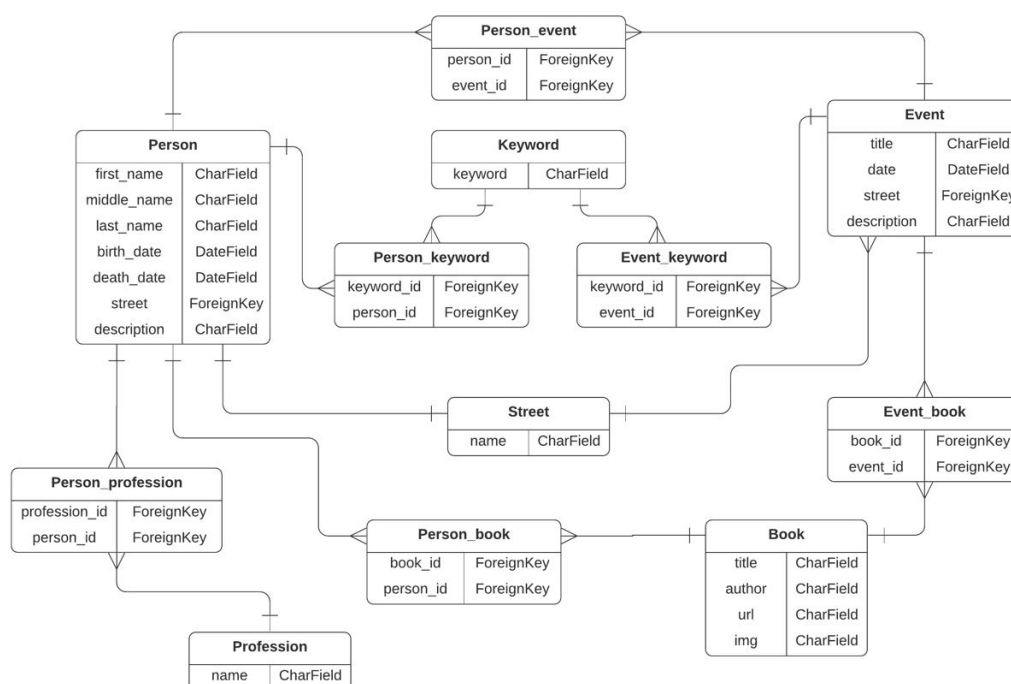


Рисунок 14 – Конечный вариант схемы базы данных

Сущности `Person_Event`, `Person_keyword`, `Event_keyword`, `Person_book`, `Event_book`, `Person_profession` являются промежуточными и помогают избежать повторения данных. Они содержат только внешние ключи сущностей, которые соединяют.

Изменение схемы базы данных повлекло переработку созданной части.

4. Оценка разработанной части.

После окончания первого прототипа необходимо было оценить проделанную работу, чтобы отметить недостающие детали, требуемые исправления.

Второй прототип

1. Доработка проекта.

На этапе доработки проекта были внесены следующие изменения:

- увеличение размера шрифта, так как текущий не устраивал заказчика,
- реорганизация структуры страниц с информацией о конкретном событии или персоне.

2. Добавление панели администратора.

Для заполнения информацией микросервис был дополнен панелью администратора (`Django admin panel`). Она представляет собой веб-интерфейс для добавления, редактирования и удаления записей в базе данных приложения. Также есть возможность добавлять пользователей или группы пользователей для предоставления доступа. В данном случае под пользователем подразумевается администратор, так как система не предусматривает других ролей. Панель администратора изображена на рисунке 15.

Она настроена таким образом, что при заполнении события или персоны можно сразу указать книги, теги и другие параметры создаваемого элемента. Это ускорит и облегчит работу администраторов.

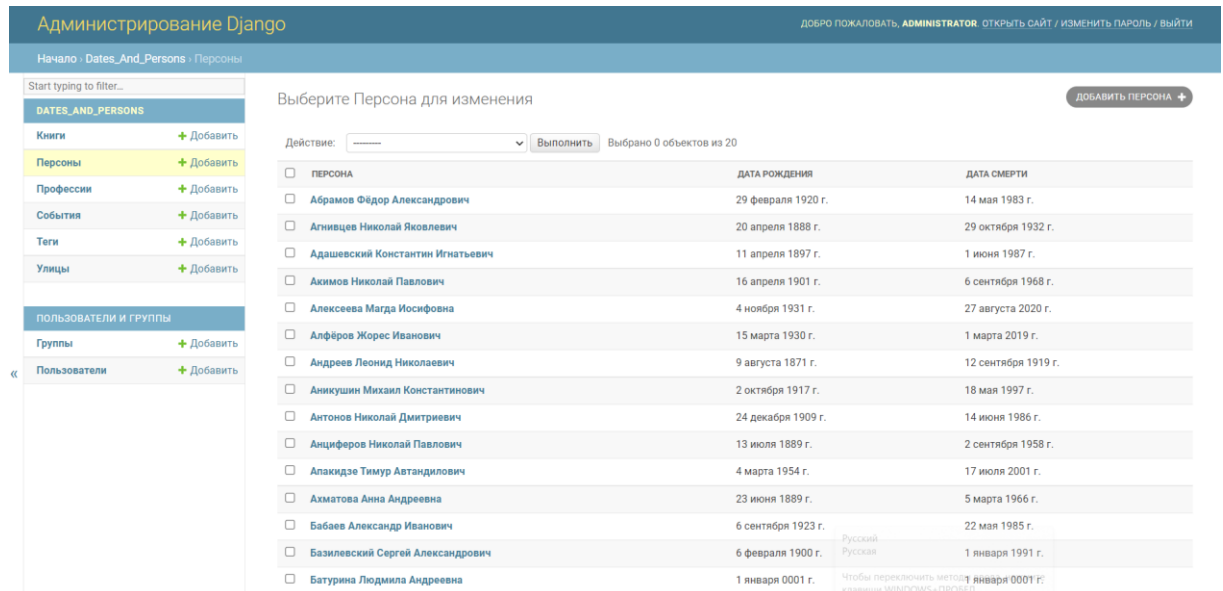


Рисунок 15 –Панель администрирования Django

3. Заполнение данными.

На данном этапе было выполнено заполнение базы данных данными с существующего сайта.

4. Оценка внесенных изменений и дополнений.

В конце работы над вторым прототипом требовалось повторно оценить текущий вариант. Новых требований от заказчика не поступило.

Третий прототип

1. Разработка API.

На данном этапе необходимо разработать API (Application Programming Interface) для того, чтобы в дальнейшем была возможность написать фронтенд не на Django, а с помощью других инструментов.

API – набор методов, который обеспечивает взаимодействие одной программы с другой [18]. В REST API обмен сообщений происходит в любом формате, например, JSON или XML.

Для проекта был выбран Django Rest Framework. С помощью ссылок можно будет получать доступ к данным.

2. Оценка созданного продукта.

На данном этапе микросервис соответствовал всем требованиям и учитывал пожелания заказчика. По окончании третьего прототипа система считается завершенной.

2.2 Разработка микросервиса

Проект был целиком реализован на Django как фронтенд, так и бэкенд.

Фронтенд – клиентская часть сайта. Бэкенд – программно-аппаратная часть, работающая на сервере.

Django является фреймворком, который использует паттерн «Модель-Представление-Контроллер» (Model-View-Controller/MVC), его схема изображена на рисунке 16.

MVC – способ организации кода, который подразумевает, что вся программа делится на три слабосвязанных компоненты – модель, представление, контроллер.

Каждая компонента отвечает за свою часть работы.

Модель отвечает за взаимодействие с базами данных, описывает, в каком виде хранятся данные, и как их нужно извлекать. Представление отвечает за то, как эти данные будут показаны пользователю, и как он будет взаимодействовать с ними. Контроллер отвечает за связь между представлением и моделью, а также реагирует на различные действия пользователя.



Рисунок 16 – Схема «Модель-Представление-Контроллер»

Для разработки проекта также были использованы следующие инструменты:

База данных SQLite3.

На сегодняшний день наиболее широко используемой системой управления базами данных является SQLite. Это однофайловая система баз данных, т.е. все таблицы хранятся только в одном файле. Здесь нет промежуточного серверного процесса. База данных может быть интегрирована непосредственно в приложение. Поэтому она предоставляет библиотеку и простой в использовании интерфейс программирования.

SQLite была выбрана по следующим причинам:

- поставляется с Django,
- легковесна,
- перед использованием не требуется сложная настройка,
- не нужны дополнительные компоненты,
- принцип «минимального полного набора»,
- подходит для данного проекта.

Django Rest Framework (DRF).

Django Rest Framework – инструмент, который позволяет создавать API сайта для удаленного взаимодействия с ним. Его преимущество не ограничивается довольно простым созданием API, оно также состоит в высокоскоростной работе, поэтому DRF может быть использован в проектах с высокой нагрузкой. Благодаря разработке API на Django Rest Framework получаем унифицированный программный код, который будет понятен большинству Django разработчиков.

Docker и Docker-compose для контейнеризации.

Docker – платформа для запуска, развертывания, управления приложений в контейнерах [24].

Преимущества использования Docker:

- приложения запускаются в изолированной среде,

- легко запускать приложения на разных серверах,
- все зависимости приложений устанавливаются внутри контейнеров,
- легко масштабировать путем увеличения количества контейнеров,
- удобно использовать в процессе разработки сайтов и приложений.

Хоть в данной работе разрабатывается только один сервис, для возможности дальнейшего расширения было принято решение создать также файл Docker-compose для того, чтобы другие сервисы можно было переписать и легко запускать.

Docker-compose – инструмент, позволяющий запускать одновременно несколько контейнеров с помощью одной команды [15]. В файле Docker-compose можно описать множество различных образов и настройки к ним.


Преимущества Docker-compose:

- декларативный подход к созданию контейнеров,
- все необходимые контейнеры запускаются одной командой,
- автоматическое создание необходимых образов на основании Dockerfile каждого приложения,
- автоматическое создание изолированной сети для взаимодействия контейнеров.

Разработка микросервиса.

При разработке был создан следующий Dockerfile, в котором описаны характеристики подключаемого образа (рисунок 17).

```

 Dockerfile > ...
1 FROM python:3.10.4-alpine
2
3 ENV PYTHONUNBUFFERED 1
4 ENV PYTHONDONTWRITEBYTECODE 1
5
6 WORKDIR /app/calendar
7
8 COPY requirements.txt .
9
10 RUN apk update && apk add postgresql-dev gcc python3-dev musl-dev \
11     && pip install --upgrade pip \
12     && pip install -r requirements.txt

```

Рисунок 17 – Dockerfile

Файл `docker-compose.yml` содержит только один сервис в связи с тем, что и фронтенд и бэкенд написаны с помощью Django (рисунок 18). В него строен одновременный запуск, ситуация аналогична с базой данных.

```

🐳 docker-compose.yml
1  version: '3.7'
2
3  services:
4    calendar:
5      build: .
6      command: sh -c "python manage.py runserver 0.0.0.0:8000"
7      ports:
8        - 8000:8000
9      volumes:
10     - ./app/calendar
11

```

Рисунок 18 – `docker-compose.yml`

С помощью созданного файла `docker-compose.yml` и команды «`docker-compose up`» запускается сервис. Запущенный сервис представлен на рисунке 19.

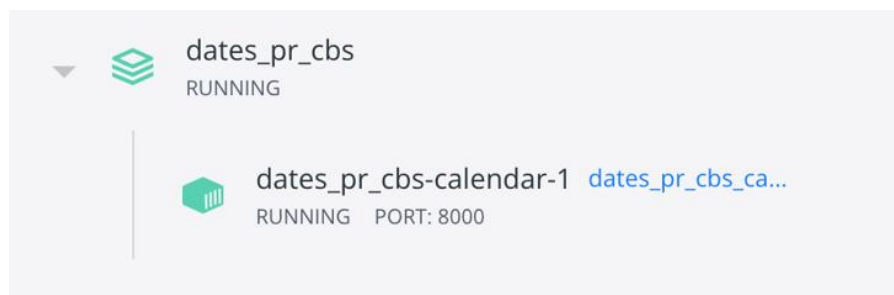


Рисунок 19 – Запущенный сервис

Первым этапом работы с приложением Django было создание моделей, которые являются основой для работы с базой данных в данном фреймворке. Модель - это поля сущности, а также правила их создания и изменения. Одна модель Django соответствует одной сущности базы данных.

На рисунке 20 показаны примеры созданных моделей:

```

4 class Street(models.Model):
5     name = models.CharField(max_length=100, verbose_name='Название')
6
7     class Meta:
8         verbose_name = 'Улица'
9         verbose_name_plural = 'Улицы'
10
11     def __str__(self) -> str:
12         return f"{self.name}"
13
14
15 class Profession(models.Model):
16     name = models.CharField(max_length=100, verbose_name='Название')
17
18     class Meta:
19         ordering = ['name']
20         verbose_name = 'Профессия'
21         verbose_name_plural = 'Профессии'
22
23     def __str__(self) -> str:
24         return f"{self.name}"
25
26

```

Рисунок 20 – Созданные модели Django

В каждой модели указаны `verbose_name` и `verbose_name_plural`, позволяющие изменить названия, которые отображаются в панели администратора. Функция «`__str__`» позволяет определить, что именно будет выводиться при запросе записи из базы данных, например, конкретное название улицы, а не номер объекта в базе данных.

Далее велась работа над созданием маршрутов Django.

Маршрутизатор получает запрос клиента и сравнивает с имеющимися у него в списке маршрутами. Контроллер работает в следующем порядке:

- вызов на запрос от клиента,
- обработка этого запроса,
- формирование запрошенных у модели данных,
- возвращение ответа в виде представления, которое заполнено этими данными.

Контроллеры могут выступать в виде контроллера функции или контроллера класса, то есть в файле `views.py` можно писать функции, либо классы, либо комбинировать их.

Все контроллеры функции должны принимать обязательный аргумент – экземпляр класса `HttpRequest request`. В нем хранятся всевозможные данные о полученном запросе, о клиенте, его браузере и так далее. В качестве ответа будет возвращен экземпляр объекта `HttpResponse`.

Маршрутизатор принимает первичный запрос клиента и сравнивает его со списком своих маршрутов. Если такой адрес есть, то он вызывает его.

Следующий шаг – работа над шаблонами.

Для того чтобы не дублировать код, в Django существуют шаблоны, которые можно рендерить. Все они находятся в папке `templates`. Каждый шаблон отвечает только за свой функционал, например, в файле `carousel.html` находится шаблон, предназначенный для карусели с обложками книг. Любая часть шаблона может быть обернута в, так называемый, блочный тег.

В качестве шаблонизатора был использован модуль `Jinja2`, который помогает при преобразовании шаблонов. Процессом рендеринга называется формирование некой страницы с данными. Он осуществляется с помощью функции `render()`, которая принимает объект запроса, путь к файлу шаблона, контекст.

Следующим этапом работы стала разработка панели администратора.

В файле `admin.py` указаны модели, которые видны в этой панели. Скриншот файла изображен на рисунке 21. Были указаны не все модели, так как у нас есть промежуточные сущности, заполнение которых происходит автоматически, при указании связи на странице редактирование персоны или события. Данный функционал был реализован с помощью создания форм в файле `forms.py`.

```

1  from django.contrib import admin
2
3  from .models import *
4  from .forms import *
5
6  admin.site.register(Book, Book_Admin)
7  admin.site.register(Event, Event_Admin)
8  admin.site.register(Person, Person_Admin)
9  admin.site.register(Street, Street_Admin)
10 admin.site.register(Profession, Profession_Admin)
11 admin.site.register(Keyword, Keyword_Admin)
12

```

Рисунок 21 – admin.py

Формы Django также позволяют изменить внешний вид различных полей, например, определить для них высоту, ширину и другое.

Работа над API.

Для начала требовалось описать методы сериализации и десериализации объектов с помощью Django Rest Framework.

Сериализация представляет собой процесс перевода данных из одной структуры в другую, которая будет удобна конечному потребителю. Десериализация – обратный процесс.

Для реализации сериализации в работе использовался класс `ModelSerializer`. Он дает возможность автоматически создать класс сериализации с полями, которые соответствуют полям модели.

`ModelSerializer` подобен `Serializer`, который является обычным классом. Его отличие заключается в следующих возможностях:

- автоматическая генерация набора полей на основе модели,
- автоматическая генерация валидаторов сериализатора,
- реализация методов `.create()` и `.update()` по умолчанию.

Класс `Meta` содержит главную часть сериализатора. В качестве модели используется соответствующая сущность. Поля, которые требуется сериализовать, должны быть установлены с помощью атрибута `fields`. В данном случае для каждого сериализатора были взяты все поля: `fields = '__all__'`.

После создания сериализаторов необходимо было создать представления для API. Класс `ModelViewSet` является наследуемым от `GenericAPIView`. В качестве атрибута `queryset` берется выборка объектов из базы данных, а в качестве `serializer_class` созданный ранее сериализатор. Например, `Street.objects.all()` вернет `QuerySet`, содержащий все объекты `Street` из базы данных. Некоторые примеры созданных представлений изображены на рисунке 22.

```
#####
# REST_FRAMEWORK
# STREET
class StreetRestFrameworkView(viewsets.ModelViewSet):
    queryset = Street.objects.all()
    serializer_class = StreetSerializer

# PROFESSION
class ProfessionRestFrameworkView(viewsets.ModelViewSet):
    queryset = Profession.objects.all()
    serializer_class = ProfessionSerializer

# BOOK
class BookRestFrameworkView(viewsets.ModelViewSet):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
```

Рисунок 22 – Представления объектов

Разработанный API основной страницы представлен на рисунке 23.

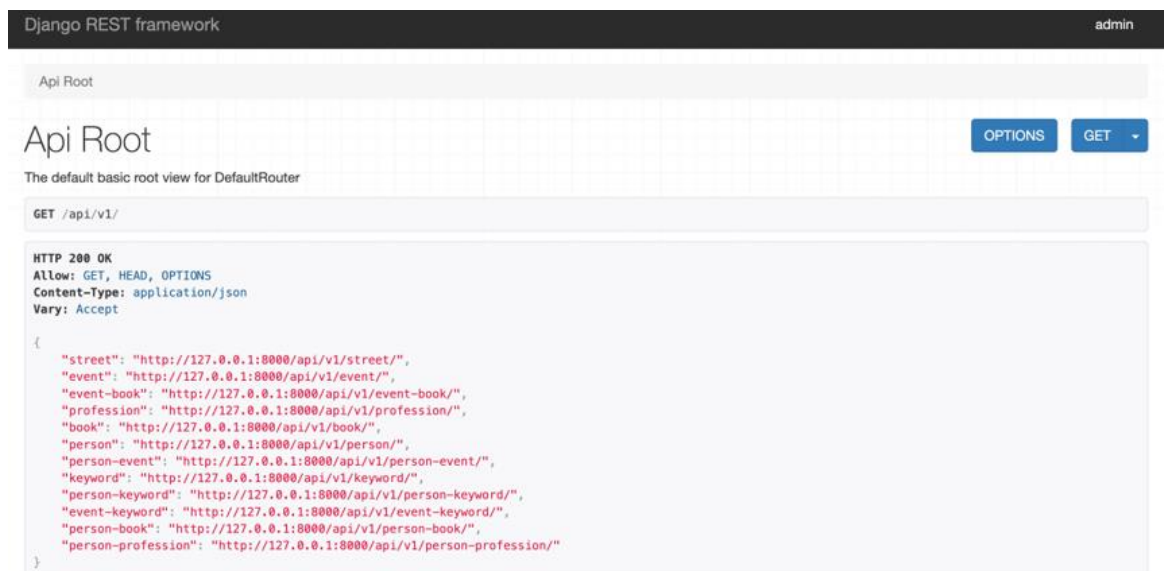


Рисунок 23 – Начальная страница API роутера Django

На данном этапе разработка проекта в рамках выпускной квалификационной работы считается завершенной.

2.3 Выводы к главе II

В главе были описаны функциональные требования к разрабатываемому продукту и поэтапный процесс разработки микросервиса на основе выбранной спиральной модели жизненного цикла.

Спроектирована UML диаграмма, которая отражает взаимодействие с системой пользователя и администратора. А также схема базы данных, описывающая архитектуру базы данных.

Осуществлен и обоснован выбор инструментов для разработки микросервиса:

- база данных SQLite3,
- Django Rest Framework для API,
- Docker и Docker-compose для контейнеризации.

Также были разработаны панель администрирования для добавления, редактирования и удаления записей, API для обмена данными, Dockerfile и docker-compose.yml для контейнеризации.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы был разработан веб-базируемый микросервис «Календарь памятных дат Петроградской стороны», содержащий 2 раздела: События и Персоны.

При выполнении работы были выполнены следующие задачи:

1. Проведен анализ существующих решений, посвященных памятным датам и выявлены преимущества разрабатываемого сервиса.
2. Спроектирована архитектура микросервиса.
3. Спроектирована схема база данных.
4. Разработан docker – образ микросервиса.
5. Разработаны фронтенд и бэкенд на Django.
6. Разработан API для дальнейшей возможности переписи фронтенда сервиса с использованием других инструментов.

В результате поставленные задачи были полностью реализованы, и цель выпускной квалификационной работы достигнута.

В дальнейшем возможны доработки и совершенствование созданного микросервиса по желанию заказчика.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Архитектурные решения информационных систем : учебник / А. И. Водяхо, Л. С. Выговский, В. А. Дубенецкий, В. В. Цехановский. — 2-е изд., перераб. — Санкт-Петербург : Лань, 2022. — 356 с. — ISBN 978-5-8114-2556-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/210020> (дата обращения: 12.04.2022). — Режим доступа: для авториз. пользователей.
2. Берг, Д. Б. Модели жизненного цикла : учебное пособие / Д. Б. Берг, Е. А. Ульянова, П. В. Добряк. — Екатеринбург : УрФУ, 2014. — 74 с. — ISBN 978-5-7996-1311-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/98755> (дата обращения: 27.04.2022). — Режим доступа: для авториз. пользователей.
3. Веб-фреймворки для начинающих: простое объяснение с примерами [Электронный ресурс]: Tproger. — 2018. — URL: <https://tproger.ru/translations/web-frameworks-how-to-get-started/> (дата обращения: 07.03.2022). — Режим доступа: свободный.
4. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг. — Москва : ДМК Пресс, 2014. — 272 с. — ISBN 978-5-97060-138-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/90103> (дата обращения: 20.04.2022). — Режим доступа: для авториз. пользователей.
5. Влацкая, И. В. Проектирование и реализация прикладного программного обеспечения : учебное пособие / И. В. Влацкая, Н. А. Заельская, Н. С. Надточий. — Оренбург : ОГУ, 2015. — 118 с. — ISBN 978-5-7410-1238-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/98065> (дата обращения: 26.03.2022). — Режим доступа: для авториз. пользователей.

6. Доррер, А. Г. Управление ИТ-проектами : учебное пособие / А. Г. Доррер, М. Г. Доррер, А. А. Попов. — Красноярск : СибГУ им. академика М. Ф. Решетнёва, 2019. — 174 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/147451> (дата обращения: 02.05.2022). — Режим доступа: для авториз. пользователей.
7. Ехлаков, Ю. П. Модели и алгоритмы управления жизненным циклом программного продукта : монография / Ю. П. Ехлаков, Д. Н. Бараксанов, Е. А. Янченко. — Москва : ТУСУР, 2013. — 196 с. — ISBN 978-5-86889-661-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/110392> (дата обращения: 03.04.2022). — Режим доступа: для авториз. пользователей.
8. Календарь знаменательных дат Карелии [Электронный ресурс]: Национальная библиотека республики Карелия. — 2020. — URL: <http://library.karelia.ru/kalendar2021/index.html#> (дата обращения: 08.03.2022). — Режим доступа: свободный.
9. Календарь знаменательных и юбилейных дат МО [Электронный ресурс]: Правительство Московской области. — 2021. — URL: <https://mosreg.ru/ob-oblasti/znamenatelnye-daty> (дата обращения: 08.03.2022). — Режим доступа: свободный.
10. Календарь исторических событий и знаменательных дат г. Ижевска [Электронный ресурс]: Администрация города Ижевска. — 2010. — URL: <https://www.izh.ru/i/info/14726.html> (дата обращения: 09.03.2022). — Режим доступа: свободный.
11. Календарь МАУК «МИБС» г. Кемерово [Электронный ресурс]: библиотеки.кемеровские.рф. — 2011. — URL: <http://библиотеки.кемеровские.рф/?p=static/calendar.php&m=2&d=1> (дата обращения: 08.03.2022). — Режим доступа: свободный.

12. Календарь памятных дат СО РАН [Электронный ресурс]: Российская академия наук Сибирское отделение. – 2010. – URL: <http://www.prometeus.nsc.ru/science/calendar/> (дата обращения: 07.03.2022). – Режим доступа: свободный.
13. Кириченко, А. В. Laravel для web-разработчиков. Практическое руководство по созданию профессиональных сайтов : руководство / А. В. Кириченко, Е. В. Дубовик. — Санкт-Петербург : Наука и Техника, 2021. — 432 с. — ISBN 978-5-94387-726-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/191484> (дата обращения: 01.05.2022). — Режим доступа: для авториз. пользователей.
14. Космачева, И. М. Проектирование защищенных баз данных : учебное пособие / И. М. Космачева, Н. В. Давидюк. — Санкт-Петербург : Интермедия, 2020. — 144 с. — ISBN 978-5-4383-0191-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/161362> (дата обращения: 19.03.2022). — Режим доступа: для авториз. пользователей.
15. Кочер, П. С. Микросервисы и контейнеры Docker : руководство / П. С. Кочер ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2019. — 240 с. — ISBN 978-5-97060-739-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/123710> (дата обращения: 02.03.2022). — Режим доступа: для авториз. пользователей.
16. Кумагина, Е. А. Модели жизненного цикла и технологии проектирования программного обеспечения : учебно-методическое пособие / Е. А. Кумагина, Е. А. Неймарк. — Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2016. — 41 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL:

- <https://e.lanbook.com/book/153391> (дата обращения: 12.05.2022). — Режим доступа: для авториз. пользователей.
- 17.Лауферман, О. В. Разработка программного продукта: профессиональные стандарты, жизненный цикл, командная работа : учебное пособие / О. В. Лауферман, Н. И. Лыгина. — Новосибирск : НГТУ, 2019. — 75 с. — ISBN 978-5-7782-3893-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/152251> (дата обращения: 18.05.2022). — Режим доступа: для авториз. пользователей.
- 18.Лоре, А. Проектирование веб-API : руководство / А. Лоре ; перевод с английского Д. А. Беликова. — Москва : ДМК Пресс, 2020. — 440 с. — ISBN 978-5-97060-861-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/179498> (дата обращения: 28.04.2022). — Режим доступа: для авториз. пользователей.
- 19.Микросервисная архитектура: теория и практика [Электронный ресурс]: VC.RU. — 2021. — URL: <https://vc.ru/dev/295980-mikroservisnaya-arhitektura-teoriya-i-praktika> (дата обращения: 07.03.2022). — Режим доступа: свободный.
- 20.Микросервисы (Microservices) [Электронный ресурс]: Хабр. — 2015. — URL: <https://habr.com/ru/post/249183/> (дата обращения: 07.03.2022). — Режим доступа: свободный.
- 21.Меле, А. Django 2 в примерах / А. Меле ; перевод с английского Д. В. Плотниковой. — Москва : ДМК Пресс, 2019. — 408 с. — ISBN 978-5-97060-746-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/123711> (дата обращения: 12.04.2022). — Режим доступа: для авториз. пользователей.
- 22.Персиваль, Г. Python. Разработка на основе тестирования. Повинуйся Билли-тестировщику, используя Django, Selenium и JavaScript / Г. Персиваль ; перевод с английского А. В. Логунов. — Москва : ДМК

- Пресс, 2018. — 622 с. — ISBN 978-5-97060-594-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/111440> (дата обращения: 10.05.2022). — Режим доступа: для авториз. пользователей.
23. Прокушев, Я. Е. Базы данных : учебное пособие / Я. Е. Прокушев. — 2-е изд., доп. — Санкт-Петербург : Интермедия, 2022. — 264 с. — ISBN 978-5-4383-0250-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/217925> (дата обращения: 28.04.2022). — Режим доступа: для авториз. пользователей.
24. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/131719> (дата обращения: 02.05.2022). — Режим доступа: для авториз. пользователей.
25. ТОП-10 фреймворков для веб-разработки [Электронный ресурс]: proglib. — 2019. — URL: <https://proglib.io/p/web-frameworks-2019> (дата обращения: 07.03.2022). — Режим доступа: свободный.
26. Хронологический справочник «Имена на карте Ленинградской области» [Электронный ресурс]: Проект Ленинградской областной универсальной научной библиотеки. — 2016. — URL: <http://xn-----bkcbbfljес5aacglpidjyhbmрcf2anрqіба.xn--p1ai/ru> (дата обращения: 09.03.2022). — Режим доступа: свободный.
27. Что такое API? [Электронный ресурс]: AWS. — 2022. — URL: <https://aws.amazon.com/ru/what-is/api/> (дата обращения: 02.03.2022). — Режим доступа: свободный.
28. 4 типа архитектуры программного обеспечения [Электронный ресурс]: NOP. — 2021. — URL: <https://nuancesprog.ru/p/12019/> (дата обращения: 07.04.2022). — Режим доступа: свободный.

29.10 лучших серверных фреймворков [Электронный ресурс]: Back4App. – 2022. – URL: <https://blog.back4app.com> (дата обращения: 20.03.2022). – Режим доступа: свободный.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

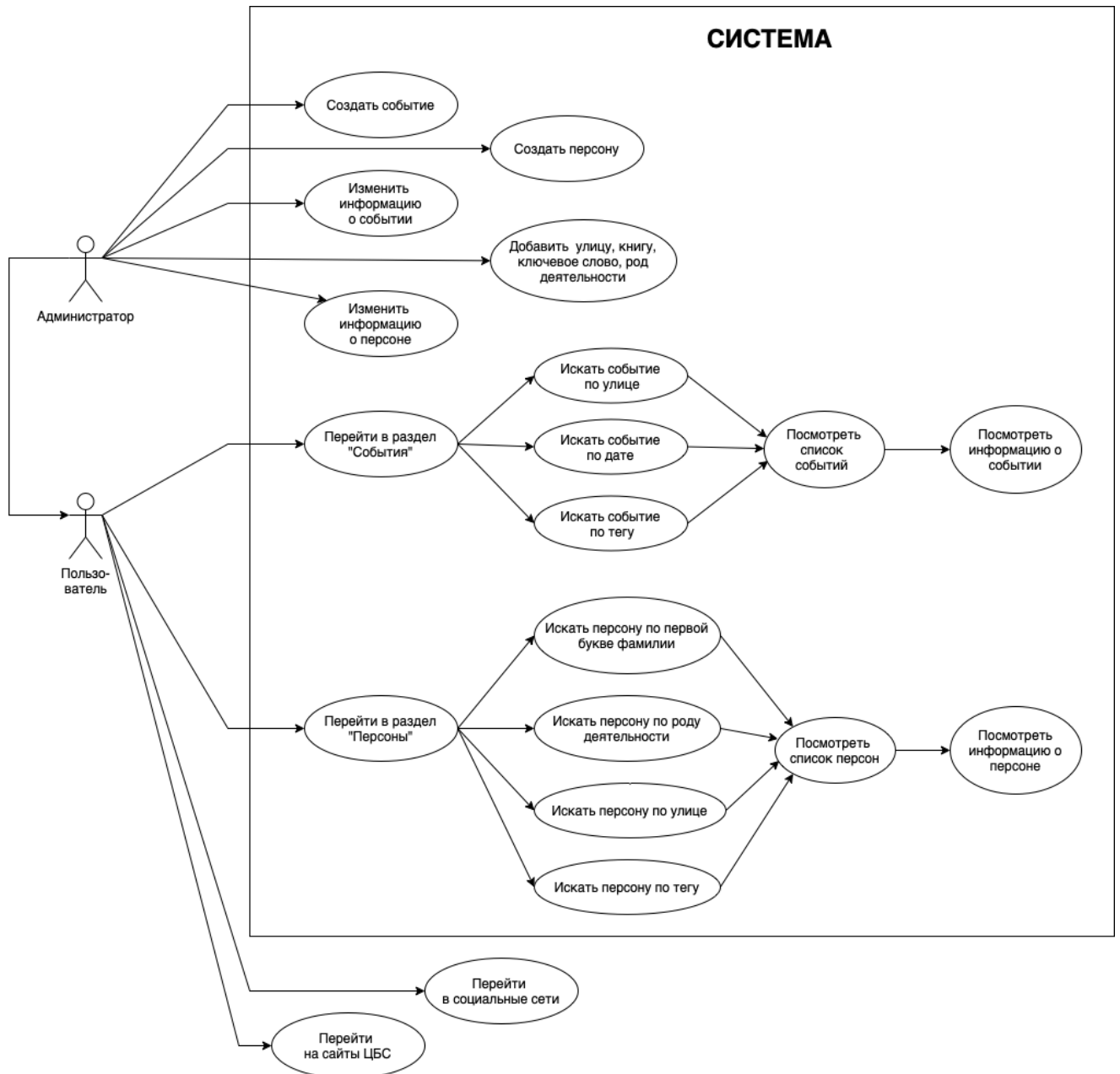


Рисунок 1 – Диаграмма прецедентов

ПРИЛОЖЕНИЕ Б

Логотип	События		Персоны
Название			
Ключевые слова		Улицы	
Поиск даты			
Календарь	Событие	Событие	
Номера страниц			
Контакты			

Рисунок 2 – Макет страницы «События»

Логотип	События			Персоны
Название				
Ключевые слова		Род деятельности	Улицы	
Алфавит для поиска				
Персона	Персона	Персона	Персона	
Номера страниц				
Контакты				

Рисунок 3 – Макет страницы «Персоны»