МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ЕДИНОГО ИНФОРМАЦИОННОГО РЕСУРСА, СОЗДАВАЕМОГО ДЛЯ НУЖД МЕСТНОГО САМОУПРАВЛЕНИЯ В РФ – ЕИР-МСУ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Тихолоза Кирилла Валерьевича

Научный руководитель
зав.каф.тех.пр.,доцент,к.ф.-м.н. _____ И. А. Батраева

Заведующий кафедрой
к.ф.-м.н., доцент _____ С. В. Миронов

Саратов 2024

ВВЕДЕНИЕ

В настоящее время развитие информационных технологий становится ключевым фактором оптимизации и повышения эффективности деятельности органов местного самоуправления в Российской Федерации. В этом контексте актуальной становится тема разработки сервера для Единого Информационного Ресурса, предназначенного для местного самоуправления (ЕИР-МСУ). Данный проект нацелен на создание централизованной системы, способствующей эффективному управлению и взаимодействию муниципальных органов, а также обеспечению граждан и предприятий всей необходимой информацией.

Заказчиками работы стали студенты юридического и экономического факультетов, они предоставили техническое задание на основе своих профессиональных знаний и требований. Проект также участвует в программе "Стартап как диплом". Разработанное приложение будет являться прототипом будущего проекта, демонстрируя его потенциал и возможности для дальнейшего развития.

Целью дипломной работы является разработка серверной части Единого информационного ресурса, создаваемого для нужд местного самоуправления в РФ – ЕИР-МСУ, на основе технического задания полученного от студентов юридического и экономического факультетов.

Предполагаемый план работы:

1. Разработка технического задания на основе идей и требований, предоставленных студентами юридического и экономического факультетов.

2. Разработка архитектуры сервера, исходя из составленного технического задания

3. Разработка методов клиент-серверного взаимодействия с учетом требований к сохранности и безопасности данных при их передаче по сети.

4. Тестирование разработанного ресурса в целом на соответствие техническим требованиям и нормам законодательства

1 Средства и технологии, используемые для разработки приложения

1.1 REST архитектура

REST (Representational State Transfer) - это стиль архитектуры для веб-приложений, где данные рассматриваются как ресурсы с уникальными адресами, доступными через стандартные методы HTTP. REST использует без состояния и предоставляет однородный интерфейс, облегчая взаимодействие между клиентом и сервером.

Ключевые концепции и принципы REST:

1. **Клиент-Сервер**

   Клиенты — это пользователи интерфейса, представляющего интерфейс пользователя, или другие приложения, использующие данные и услуги от сервера.

   Серверы управляют ресурсами и предоставляют их клиентам. Разделение на клиентов и серверы облегчает масштабирование, изменение и поддержку систем.

2. **Отсутствие Состояния**

   Каждый запрос от клиента к серверу должен содержать всю необходимую информацию для понимания и выполнения запроса. Сервер не должен сохранять состояние между запросами от одного клиента к другому.

3. **Кешируемость**

   Ответы сервера могут быть помечены как кэшируемые или некэшируемые. Клиенты могут использовать кэширование для улучшения производительности, уменьшая нагрузку на сервер.

4. **Единый Интерфейс**

   Унифицированный интерфейс обеспечивает стандартизированный способ взаимодействия между клиентами и серверами. Он включает в себя:

   Каждый ресурс должен быть идентифицирован уникальным URI.

   Клиенты и серверы обмениваются представлениями ресурсов, например, в формате JSON или XML.

   Каждое сообщение содержит всю необходимую информацию для его понимания.

   Гиперссылки используются для связи между компонентами системы, позволяя клиентам навигировать и взаимодействовать с ресурсами.

5. **Слои**

Архитектура REST может быть построена в виде многослойной системы, где каждый слой выполняет определенные функции. Это облегчает масштабирование и изменение, а также повышает устойчивость системы.

6. **Код по Требованию**
   Этот принцип является необязательным и предполагает, что клиенты могут загружать и выполнять код от сервера по запросу.

1.2 Протокол HTTP

HTTP — это протокол передачи данных, используемый для передачи информации в Интернете. В HTTP есть несколько основных методов и коды ответа, которые используются для взаимодействия между клиентом (обычно браузером) и сервером.

Основные методы HTTP:

— **GET:** Запрашивает представление ресурса. Не должен изменять состояние сервера.
— **POST:** Отправляет данные на сервер для создания нового ресурса.
— **DELETE:** Удаляет указанный ресурс.
— **HEAD:** Запрашивает заголовки ресурса, без его тела. Полезно для проверки существования ресурса.
— **OPTIONS:** Запрашивает поддерживаемые методы для ресурса или сервера.

Основные коды ответа HTTP:

— **1xx** (Информационные)
— **2xx** (Успешные)
— **3xx** (Перенаправление)
— **4xx** (Ошибка клиента)
— **5xx** (Ошибка сервера)

Протокол HTTPS — это расширение протокола HTTP, обеспечивающее безопасную передачу данных через компьютерные сети. Протокол HTTPS защищает конфиденциальность и целостность данных между устройствами пользователя и сервером посредством использования шифрования с помощью SSL или его современного преемника TLS.

1.3 Передача информации с помощью JSON Web Token (JWT)

Одним из наиболее популярных и безопасных способов реализации авторизации в веб-приложениях является использование JSON Web Token (JWT)

JWT – это стандарт, для создания токенов, которые могут быть использованы для авторизации и информационного обмена.

Рисунок 1 – Пример JWT-токена

JWT-токен состоит из трех частей, разделенных точками:

— **Заголовок:** содержит информацию о типе токена и алгоритме подписи.

— **Полезная нагрузка:** содержит утверждения, которые представляют собой набор информации о пользователе и других данных.

— **Подпись:** используется для верификации целостности токена и удостоверения подлинности источника.

Одной из особенностей JWT-токена является прозрачность данных, которые содержатся в полезной нагрузке токена. В отличие от некоторых других методов авторизации, данные внутри JWT не зашифрованы по умолчанию. Это означает, что любой, кто имеет доступ к JWT, может просмотреть его содержимое, просто декодировав его из формата base64Url.

Рисунок 2 – Раскодированный JWT-токен

1.4 Протокол OAuth 2.0

OAuth 2.0 — это протокол для авторизации, который предоставляет стандартизированный способ делегирования доступа к ресурсам от одной стороны (например, пользователя) третьей стороне (например, приложению) без необходимости передачи логина и пароля. OAuth 2.0 часто используется для обеспечения безопасного доступа к API и веб-ресурсам.

1.5 Средства для разработки

Java — это мощный и универсальный язык программирования, который обеспечивает объектно-ориентированный подход и высокую производительность благодаря компиляции в байт-код, исполняемый на виртуальной машине Java (JVM). Для программиста Java предоставляет обширную стандартную биб-

лиотеку и множество фреймворков, упрощающих разработку масштабируемых, надежных и платформо-независимых приложений.

Java Spring — это популярный и мощный фреймворк для разработки приложений на языке программирования Java. Spring обеспечивает комплексное решение для создания масштабируемых, надежных и эффективных Java-приложений.

Spring Boot — это проект в рамках экосистемы Spring, предоставляющий простые средства для создания самостоятельных, готовых к использованию приложений на базе Spring. Он упрощает конфигурацию и развертывание приложений, предоставляя множество конвенций по умолчанию.

Spring Security — это фреймворк для обеспечения безопасности приложений на платформе Java. Он предоставляет обширные инструменты для аутентификации (проверки подлинности) и авторизации (контроля доступа) в веб-приложениях, микросервисах и других типах приложений.

Hibernate — это фреймворк для работы с базами данных в Java приложениях. Он предоставляет инструменты для объектно-реляционного отображения (ORM), что позволяет разработчикам взаимодействовать с базой данных, используя объектно-ориентированный подход.

JPA Repository - это высокоуровневый абстрактный слой, предоставляющий набор CRUD операций и методы для работы с данными в реляционных базах данных, используя возможности JPA.

Hibernate как раз одна из наиболее популярных реализаций JPA

PostgreSQL является мощной и гибкой системой управления базами данных, которая подходит для широкого спектра приложений. Её открытый исходный код, соответствие стандарту SQL и уникальные возможности делают её привлекательным выбором для разработчиков. Благодаря активному сообществу и постоянному развитию, PostgreSQL продолжает улучшаться и оставаться одной из ведущих систем управления базами данных.

Git — это распределённая система контроля версий, созданная для управления проектами с быстро изменяющимся исходным кодом.

2 Разработка приложения

2.1 Составление технического задания

Перед началом разработки приложения было составлено техническое задание, чтобы определить, какой функциональностью должно обладать приложение. Для создания технического задания были проведены беседы со студентами юридического и экономического факультетов.

По итогам бесед была сформулирована основная идея приложения. Необходимо создать сервис для удобного документооборота муниципалитетов с возможностью быстрого поиска законопроектов. Также в приложении должна быть функция, позволяющая обычным жителям получать детальную информацию о проектах конкретных депутатов или о работе Думы и комиссий в целом. Эта функция должна включать просмотр и поиск предложенных законопроектов и принятых решений, просмотр хода реализации проектов, а также возможность ознакомления с недавними изменениями и публикациями.

Необходимо также обеспечить безопасность работы приложения.

После создания проекта нужно будет протестировать его на Молодежном парламенте при Саратовской городской Думе.

По итогу необходимо разработать:

1. Клиентскую часть приложения.
2. Сервис авторизации и регистрации.
3. Сервис для работы с документами.
4. Поисковую систему для документов.

В соответствии с разработанным техническим заданием моя задача заключается в разработке сервиса авторизации и регистрации, сервиса работы с документами и обеспечение их безопасности.

2.2 Архитектура приложения

Использование концепции REST следует предпочесть из-за её простоты и легкости интеграции, что позволяет создавать масштабируемые и легко поддерживаемые веб-сервисы.

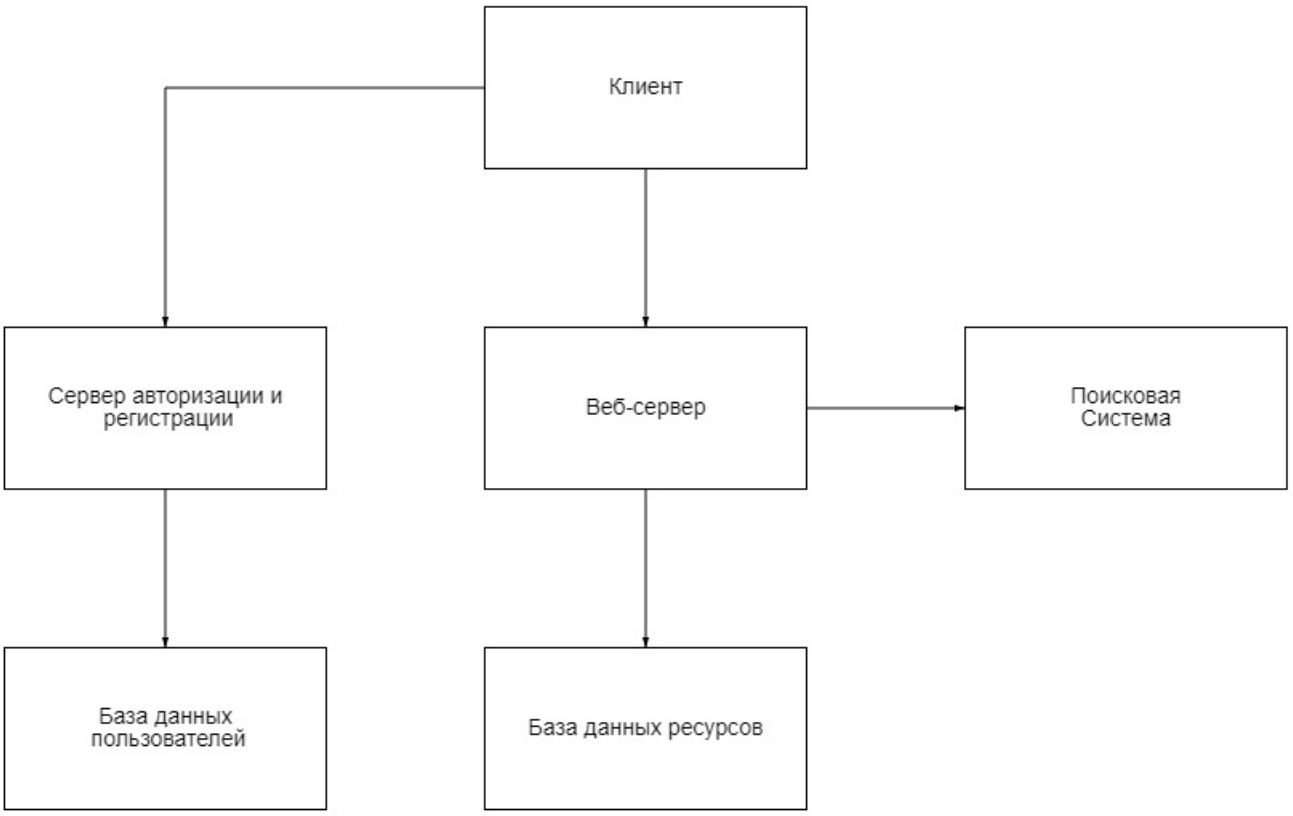

Рисунок 3 – Архитектура приложения

Клиент - RESTful веб-клиент, который представляет собой веб-сайт для общения с сервером

Веб-Сервер - RESTful сервер, в контексте веб-приложения, предоставляет ресурсы и обрабатывает запросы от клиентов.

Сервер авторизации и регистрации - Сервер авторизации и регистрации в контексте RESTful архитектуры выполняет специализированные функции, связанные с проверкой подлинности пользователей и выдачу им соответствующих прав, а так же с добавлением новых пользователей в систему.

База данных пользователей - является хранилищем информации о пользователях, она играет важную роль в аутентификации, авторизации и общем управлении пользователями.

Поисковой движок - сервис в RESTful архитектуре предоставляющий возможность клиентам выполнять поисковые запросы и получать результаты поиска через стандартные HTTP-методы.

База данных ресурсов - в контексте RESTful архитектуры является хранилищем данных ресурсов, которые использует Веб-Сервер.

2.3 Сервер авторизации и регистрации

Основными задачами для данного микросервиса являются:

— Добавление пользователей в систему
— Создание JWT-токена, когда пользователь авторизируется в системе
— Обеспечение защиты информации о пользователе

Функциональные возможности сервера авторизации и регистрации

1. Обработчик запроса на пути /registrate добавляет пользователей в систему.
2. Обработчик запроса на пути /auth проверяет информацию о пользователе и на ее основе генерировать access и refresh токены.
3. Обработчик запроса на пути /refresh обновляет access и refresh токены для безопасности системы и удобства пользователя.
4. Обработчик запроса на пути /oauth перенаправляет пользователя на авторизацию через VK API (сервис ВКонтакте).
5. Обработчик запроса на пути /callback принимает токен доступа от VK API для возможности взаимодействия с пользователем, авторизованным через VK API, и регистрирует его в системе “Единого Информационного Ресурса”.

2.4 Веб-Сервер

Веб-сервер отвечает за прием, обработку и выполнение запросов от клиентов. В данном приложении он отвечает за все взаимодействия пользователя с проектами и решениями, такие как создание проектов, проведение проектов в решения, удаление проектов или решений, а также отображение и сортировку проектов или решений. Веб-сервер также отвечает за отображение хода реализации законопроектов. Он генерирует новостную ленту, в которой отображается прогресс принятия законопроектов, а также показываются последние предложенные проекты или принятые решения.

В предметной области в которой будет работать данное приложение есть два определения, понимание которых поможет лучше разобраться в работе приложения:

— **Проект** — это предложенный нормативный акт, который должен пройти процесс обсуждения и утверждения, чтобы стать законом.

— **Решение** — это итоговый акт, принимаемый законодательным органом, который определяет исход голосования по законопроекту или другим вопросам.

Данное приложение как раз направлено для удобного взаимодействия с проектами и решениями.

Функциональные возможности веб-сервера

Веб-сервер отвечает за работу с законопроектами. Создана связь с микросервисом поисковика. Созданы такие обработчики запросов:

Обработчики запросов доступные только для модераторов:

1. Обработчик запроса на пути /documents/download добавляет законопроект в систему.
2. Обработчик запроса на пути /documents/delete удаляет законопроект из системы.
3. Обработчик запроса на пути /documents/provide проводит законопроект в решение.
4. Обработчик запроса на пути /addProjectStep добавляет один шаг в ход реализации выбранного законопроекта.

Обработчики запросов доступные для обычных пользователей:

1. Обработчик запроса на пути /resources/get/id возвращает документ законопроекта для отображения проекта в браузере.
2. Обработчик запроса на пути /resources/documents возвращает список состоящий из информации о законопроектах и решениях для формирования ленты из всех документов в системе.
3. Обработчик запроса на пути /resources/documentsByAuthor возвращает список состоящий из информации о законопроектах и решениях, отсортированных по указанному автору, для формирования ленты проектов в профиле пользователя с которыми он взаимодействовал.
4. Обработчик запроса на пути /resources/projects возвращает список состоящий из информации о законопроектах для формирования ленты из законопроектов.
5. Обработчик запроса на пути /resources/solutions возвращает список состоящий из информации о решениях для формирования ленты из решений.

6. Обработчик запроса на пути /resources/news возвращает список состоящий из информации о всех шагах реализации, отсортированных по дате, для формирования ленты новостей.
7. Обработчик запроса на пути /getProjectSteps возвращает список состоящий из информации о шагах реализации, связанных с конкретным законопроектом, для отображения хода реализации.
8. Обработчик запроса на пути /search/projects необходим для перенаправления запроса на поиск законопроектов в микросервис поисковика и формирования из полученного ответа информации о законопроектах и ее передача на клиентскую часть приложения.
9. Обработчик запроса на пути /search/solution необходим для перенаправления запроса на поиск решений в микросервис поисковика и формирования из полученного ответа информации о законопроектах и ее передача на клиентскую часть приложения.
10. Обработчик запроса на пути /search/documents необходим для перенаправления запроса на поиск законопроектов и решений в микросервис поисковика и формирования из полученного ответа информации о законопроектах и ее передача на клиентскую часть приложения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы был получен ценный опыт командной работы со студентами других факультетов. На основе их требований было разработано техническое задание для определения основного функционала приложения.

На основании полученных данных была создана архитектура, обеспечивающая эффективную работу приложения в данной предметной области. В результате было разработано и протестировано приложение для Единого информационного ресурса, предназначенного для нужд местного самоуправления в РФ – ЕИР-МС. Так же был получен опыт создания стартапа благодаря участию в программе "Стартап как диплом".