



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

---

---

**Институт информационных технологий (ИТ)**  
**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВАЯ РАБОТА**

по дисциплине: Разработка серверных частей интернет-ресурсов  
по профилю: Разработка программных продуктов и проектирование информационных систем  
направления профессиональной подготовки: Программная инженерия (09.03.04)

Тема: «Веб-сервис версионного контроля»

Студент: Московка Артём Александрович

Группа: ИКБО-20-19

Работа представлена к защите 23.11.21 /Московка А.А./  
(подпись и ф.и.о. студента)

Руководитель: Лобанов Александр Анатольевич, доцент кафедры ИиППО

Работа допущена к защите \_\_\_\_\_ (дата) /Лобанов А.А./  
(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: \_\_\_\_\_

\_\_\_\_\_/\_\_\_\_\_  
\_\_\_\_\_/\_\_\_\_\_

(подписи, дата, ф.и.о., должность, звание, уч. Степень двух преподавателей, принявших  
защиту)



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

---

---

**Институт информационных технологий (ИТ)**  
**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ЗАДАНИЕ**  
**на выполнение курсовой работы**

по дисциплине: Разработка серверных частей интернет-ресурсов  
по профилю: Разработка программных продуктов и проектирование информационных систем  
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Московка Артём Александрович

Группа: ИКБО-20-19

Срок представления к защите: 23.11.2021

Руководитель: Лобанов Александр Анатольевич, доцент кафедры ИиППО

**Тема:** «Веб-сервис версионного контроля»

**Исходные данные:** используемые технологии: HTML5, CSS3, JetBrains IDE, SQL СУБД, наличие: межстраничной навигации, внешнего вида страниц, соответствующего современным стандартам веб-разработки, использование паттерна проектирования MVC. Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

1. Провести анализ предметной области разрабатываемого веб-приложения. 2. Обосновать выбор технологий разработки веб-приложения. 3. Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования. 4. Реализовать слой серверной логики веб-приложения с применением выбранной технологии. 5. Реализовать слой логики базы данных. 6. Разработать слой клиентского представления веб-приложения 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: \_\_\_\_\_ /Р. Г. Болбаков/, « \_\_\_\_\_ » \_\_\_\_\_ 2021 г.

Задание на КР выдал: \_\_\_\_\_ /А.А Лобанов/, « \_\_\_\_\_ » \_\_\_\_\_ 2021 г.

Задание на КР получил: \_\_\_\_\_ /А.А. Московка/, « \_\_\_\_\_ » \_\_\_\_\_ 2021 г.

## Глоссарий

1. API — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.
2. Bitbucket — веб-сервис для хостинга проектов и их совместной разработки, основанный на системах контроля версий Mercurial и Git.
3. CI/CD или CICD — это комбинация непрерывной интеграции и непрерывного развертывания программного обеспечения в процессе разработки.
4. Clean Architecture — паттерн проектирования веб-приложений, обеспечивающий возможность тестировать модули, независимый от UI и независимый от БД, внешних фреймворков и библиотек.
5. CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание, чтение, модификация, удаление.
6. Dart — язык программирования, созданный Google. Dart позиционируется в качестве замены/альтернативы JavaScript.
7. DDD (Domain-Driven Design) — это набор принципов и схем, направленных на создание оптимальных систем объектов. Сводится к созданию программных абстракций, которые называются моделями предметных областей. В эти модели входит бизнес-логика, устанавливающая связь между реальными условиями области применения продукта и кодом.
8. Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.
9. Flutter — комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, а также веб-приложений с использованием языка

- программирования Dart, разработанный и развиваемый корпорацией Google.
- 10.GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.
  - 11.GitLab — веб-инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, CI/CD пайплайном и другими функциями.
  - 12.Google — американская транснациональная корпорация в составе холдинга Alphabet, инвестирующая в интернет-поиск, облачные вычисления и рекламные технологии.
  - 13.Google Chrome — браузер, разрабатываемый компанией Google на основе свободного браузера Chromium и движка Blink.
  - 14.IDE – Интегрированная среда разработки – комплекс программных средств, используемый программистами для разработки программного обеспечения.
  - 15.IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.
  - 16.JetBrains — международная компания, которая разрабатывает инструменты для разработки на языках Java, Kotlin, C#, F#, C++, Ruby, Python, PHP, JavaScript и многих других, а также средства командной работы.
  - 17.Lombok — это плагин компилятора, который добавляет в Java новые «ключевые слова» и превращает аннотации в Java-код, уменьшая усилия на разработку и обеспечивая некоторую дополнительную функциональность.
  - 18.MongoDB — документоориентированная система управления базами

данных, не требующая описания схемы таблиц.

- 19.MVC (Model-View-Controller) — схема разделения данных приложения, и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.
- 20.NoSQL — обозначение широкого класса разнородных систем управления базами данных, появившихся в конце 2000-х — начале 2010-х годов и существенно отличающихся от традиционных реляционных СУБД с доступом к данным средствами языка SQL.
- 21.Postman — это HTTP-клиент для тестирования API.
- 22.REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
- 23.SQL — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.
- 24.Subversion (SVN) — свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet.
- 25.VCS — Система управления версиями — программное обеспечение для облегчения работы с изменяющейся информацией.
- 26.Бойлерплейт — это фрагменты кода, которые повторяются в нескольких местах с небольшими вариациями или без них.

## Оглавление

Глоссарий .....	3
Введение .....	7
1. Основная часть .....	8
1.1. Анализ предметной области .....	8
1.2. Выбор и обоснование технологий .....	9
1.3. Разработка архитектуры приложения на основе выбранного паттерна.....	13
1.4. Разработка серверной части интернет-ресурса .....	17
1.5. Разработка клиентской части интернет-ресурса .....	20
1.6. Тестирование.....	22
Заключение.....	25
2. Список литературы .....	26

## **Введение**

Целью данной курсовой работы является исследование механизма работы систем контроля версий и последующая разработка собственного прототипа. Существующие системы контроля версий достаточно развиты и не требуют совершенствования, поэтому и пользуются такой популярностью в обществе разработчиков.

Актуальность данной исследовательской и практической работы крайне высока, поскольку VCS повсеместно используются, поэтому важно и нужно понимать, как они устроены и работают.

Объектом и предметом исследования данной курсовой работы является веб-приложение, по типу GitHub или GitLab, обладающее возможностью сохранения файлов и их версий с последующей возможностью загрузки любой желаемой версии приложения, также обеспечивающее защиту от несанкционированного доступа к файлам.

Предполагаемым результатом является веб-сервис, позволяющий сохранить файл с указанием актуальной версии, загрузить новую версию уже измененного файла и скачать любую из загруженных версий, кроме того, имеющий в себе механизм регистрации и авторизации с целью защиты данных от других пользователей.

## **1. Основная часть**

### **1.1. Анализ предметной области**

До начала 21 века системы контроля версий не были такими удобными и надежными, какими они являются сейчас. В 2000 году появился готовый программный продукт Subversion или SVN, позволяющий отслеживать версии файлов – это уже был большой шаг, но позже появились распределенные системы контроля версий, которыми мы и пользуемся в настоящее время.

Самым популярным существующим сервисом является GitHub, разработанный Томом Престоном-Вернером, Скоттом Чаконом, Крисом Ванстратом и П. Дж. Хьеттом в 2008 году. Данный сервис позволяет отправлять файл на сервер, просматривать содержимое текстовых файлов, сливать ветки и устранять конфликты между версиями файлов.

К другим примерам актуальных сервисов можно причислить GitLab и BitBucket, обладающие похожим функционалом, что и вышеупомянутый сервис, но имеющие функциональные отличия. Например, GitLab имеет удобную систему CI/CD, позволяющую автоматически развертывать обновления на сервере, чем не может похвастаться GitHub при тех же условиях использования. А BitBucket позволяет загружать большие файлы без существенного увеличения веса истории коммитов.

Из вышесказанного можно сделать вывод о том, что в разрабатываемом веб-приложении должна присутствовать возможность загрузки файлов с различными версиями, отслеживания версий файлов и скачивания любой версии на выбор.

### **Выводы к первой главе**

В результате анализа предметной области было сказано про основные функциональные составляющие имеющихся решений и были представлены основные требования к создаваемому веб-ресурсу.



## 1.2.Выбор и обоснование технологий

Выбор на архитектуру проектирования приложения MVC пал по причине того, что данная архитектура удобно позволяет реализовать требуемый функционал, не перегружена обработкой сложной бизнес-логики, на которую нацелена архитектура DDD, и подходит лучше, чем Clean Architecture, в которой играют большую роль unit-тесты, отсутствующие в данной работе.

С целью успешного выполнения поставленных в данном проекте задач, а еще по той причине, что разрабатывается именно web-сервис, использовалась архитектура проектирования REST, в которой происходит взаимодействие с данными, которые находятся в запросах, отправленных или полученных клиентом от сервера. Большим плюсом является то, что данная архитектура рассматривается в курсе по дисциплине «Интерфейсы прикладного программирования» данного семестра.

Поскольку неизвестно, сколько файлов, какие имена и какой объем файлов будет загружен пользователем, было принято решение использовать нереляционные базы данных, в частности систему нереляционных баз данных MongoDB, в которой имеется все необходимое для записи, и получения данных, внесенных в хранилище.

Клиентская часть разработана на фреймворке Flutter, он выбран из-за нарастающей популярности и общей скорости разработки продукта. Он использует собственный движок для отображения элементов на странице, поэтому и может обеспечить быстрое действие. Его разработчиком является компания Google, которая разработала язык программирования Dart, для которого и написан данный фреймворк.

Основным языком программирования при построении архитектуры и серверной части приложения выбран язык Java, поскольку является наиболее знакомым вкупе с удобным и надежным фреймворком Spring для построения структуры и контроллеров взаимодействия с API. Автоматической сборкой

системы будет заниматься сборщик систем Gradle. Для развертывания веб-приложения на различных операционных системах и устройствах используется система контейнеризации Docker. Чтобы сократить объем бойлерплейт кода, была использована библиотека Lombok.

Из личных предпочтений был сделан выбор интерактивной среды разработки IntelliJ IDEA от JetBrains. Данная IDE имеет приятный внешний вид, много опций для кастомизации, поддерживает все используемые для написания курсовой работы технологии и, что не менее важно, данное ПО можно приобрести бесплатно в образовательных целях. На рисунках 1-3 приведены основные преимущества, встроенные модули, и инструменты для разработки ПО.

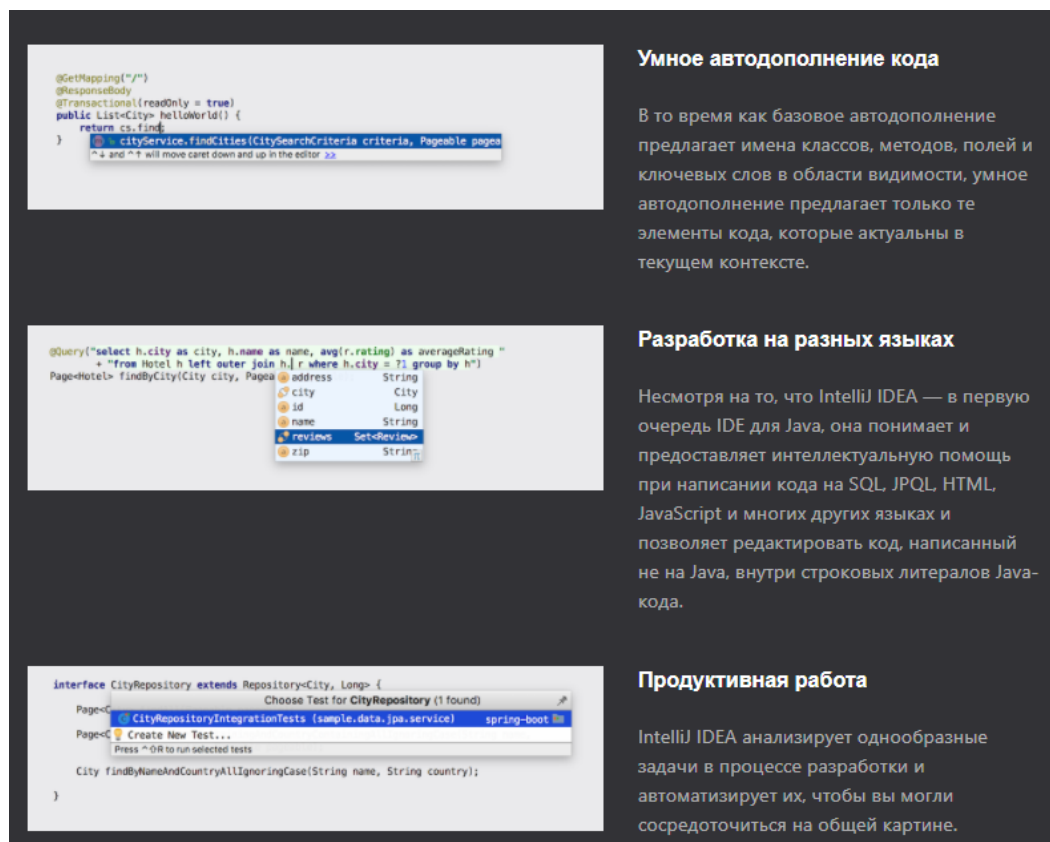


Рисунок 1 – Скриншот основных преимуществ среды IntelliJ IDEA

## Эргономичная среда

Работая над функциональностью и дизайном IDE, мы всегда думаем о том, чтобы разработчика ничто не отвлекало, и делаем все возможное, чтобы устранить или минимизировать риск потери контекста.

IntelliJ IDEA понимает, что вы хотите сделать, и автоматически вызывает соответствующие инструменты.

## Продуктивность во всем

IntelliJ IDEA не только помогает писать код в редакторе, но и упрощает работу в целом. Вы сможете легко и быстро заполнить поле, найти элемент в списке, открыть нужное окно, поменять настройки и т.д.

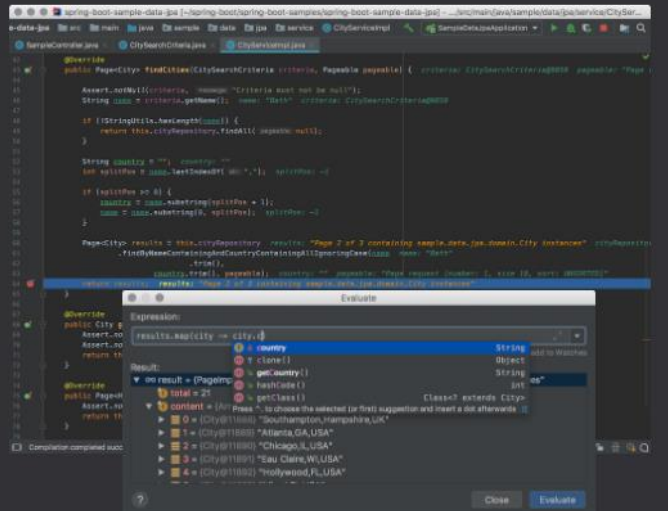


Рисунок 2 – Скриншот основных преимуществ среды IntelliJ IDEA






				
<b>Встроенные инструменты</b>	<b>Языки JVM</b>	<b>Фреймворки для разработки корпоративных приложений</b>	<b>Мобильная разработка</b>	<b>Веб-разработка</b>
Инструменты сборки	Java		Android	JavaScript
Интеграция с системами контроля версий	Kotlin		React Native	TypeScript
HTTP-клиент	Scala	Spring	Cordova	HTML и CSS
Инструменты профилирования	Groovy	Java EE	Ionic	Node.js
Декомпилятор		Jakarta EE		Angular
Покрывтие кода		Micronaut		React
Работа с базами данных и SQL		Quarkus		Vue.js
		Helidon		
		Grails		

Рисунок 3 – Скриншот встроенных инструментов и поддерживаемых фреймворков IntelliJ IDEA

## Выводы к второй главе

Результатом работы по данной главе стал ряд описанных технологий и причин, почему были выбраны и использованы именно эти технологии. Можно также упомянуть про браузер Google Chrome, в котором будет происходить работа с веб-приложением.

### 1.3.Разработка архитектуры приложения на основе выбранного паттерна

Файлы в рабочей директории проекта расположены в соответствии с принципами архитектуры MVC, что подразумевает наличие моделей (Model), страниц и их визуального отображения (View) и сервисов и контроллеров для взаимодействия пользователя с моделью данных (Controller), что можно видеть на рисунке 4.

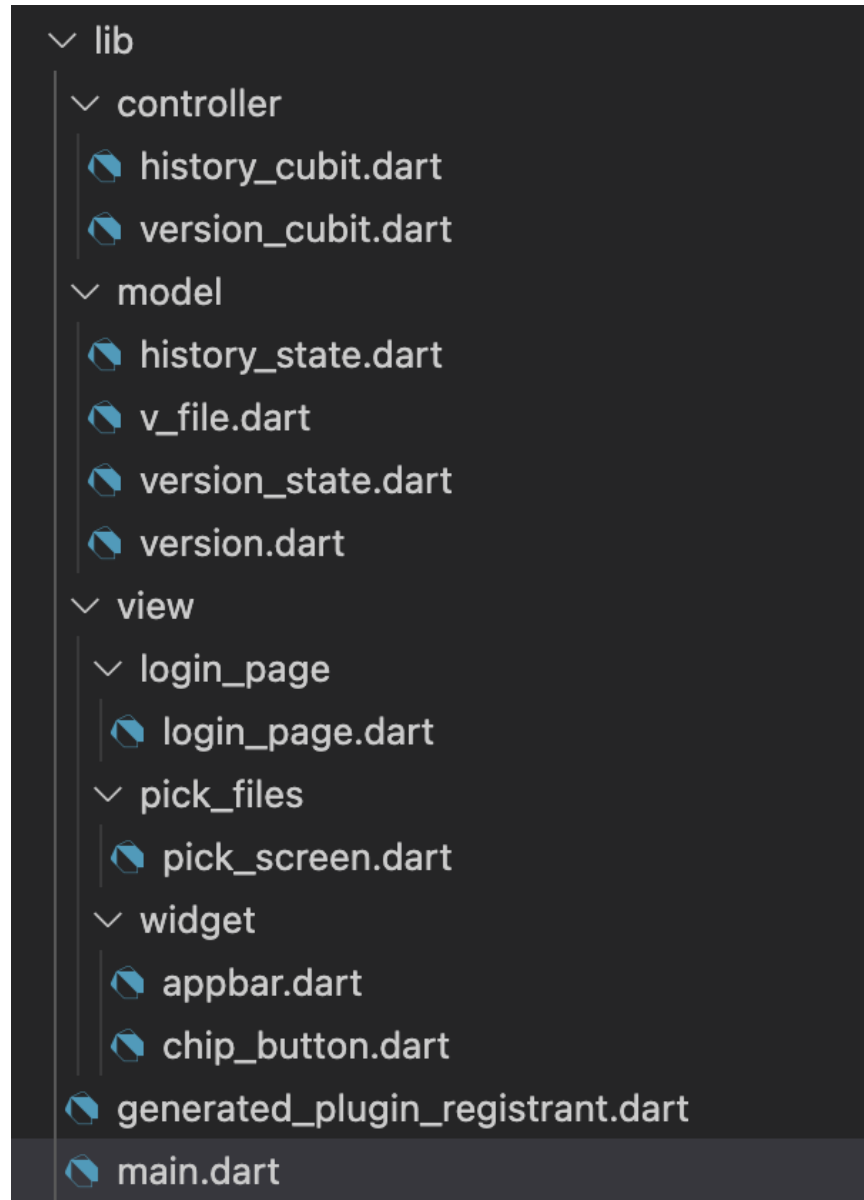


Рисунок 4 – Скриншот директория проекта в соответствии с архитектурой MVC

На представленном ниже рисунке 5 представлен класс, реализующий кубит, который хранит состояние истории версий, позволяет добавить новую версию в список версий и проверяет корректность имени версии.

```
class HistoryCubit extends Cubit<HistoryState> {
  HistoryCubit() : super(const HistoryState([]));

  void addVersion(Version v) {
    List<Version> newV = List<Version>.from(state.versions);
    newV.add(v);
    newV.sort((a, b) => b.date.compareTo(a.date));
    emit(HistoryState(newV));
  }

  bool isAllowedToSave(Version v) {
    if (v.versionName.isEmpty) return false;
    bool f = true;
    for (var v in state.versions) {
      if (v.versionName == v.versionName) f = false;
    }
    return f;
  }
}
```

Рисунок 5 – Скриншот кода кубита истории версий

Серверная часть веб-приложения построена на архитектуре проектирования RESTful API, а само API пользователей реализует контроллер, представленный на рисунке 6. Он содержит методы для реализации CRUD группы, то есть создание, чтение, обновление и удаление данных, и метод получения всех пользователей, а также авторизации конкретного пользователя на сервисе.

```

@RestController
@RequestMapping("/api")
@RequiredArgsConstructor
public class UserController {

    private final UserService userService;

    @GetMapping("/all")
    public List<User> findAllUsers() {
        return userService.findAll();
    }

    @GetMapping("/{login}")
    public User findById(@PathVariable String login) {
        return userService.findByLogin(login);
    }

    @PostMapping("/registration")
    public ResponseEntity saveUser(@RequestBody User user) {
        try {
            userService.findByLogin(user.getLogin());
        } catch (RuntimeException ex) {
            user.setRoles(Collections.singleton(Role.USER));
            userService.save(user);
            return new ResponseEntity<>(HttpStatus.OK);
        }
        throw new RuntimeException("User already exists!");
    }

    @PutMapping("/update")
    public ResponseEntity updateUser(@RequestBody User user) {
        Document document = Document.parse(user);
        update(document);
        return ResponseEntity.ok(HttpStatus.OK);
    }

    @DeleteMapping("/delete")
    public ResponseEntity deleteUser(@RequestBody User user) {
        Document document = Document.parse(user);
        delete(document);
        return ResponseEntity.ok(HttpStatus.OK);
    }
}

```

Рисунок 6 – Скриншот кода контроллера пользователей

В файле Docker-compose описана конфигурация приложения, а именно: приложение на ЯП Java, база данных MongoDB, отображенные на рисунке 7.

```
version: '3.9'

services:
  api:
    container_name: api_vcs|
    build: .
    links:
      - mongo
    ports:
      - 8080:8080

  mongo:
    container_name: mongo
    hostname: mongo
    image: mongo:4.4
    volumes:
      - ./db_data:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: secret
    ports:
      - 27017:27017
```

Рисунок 7 – Скриншот содержимого файла docker-compose

### Выводы к третьей главе

По выполнении данной главы было детально описано использование и применение паттерна проектирования и выбранных технологий для построения архитектуры приложения, продемонстрирована структура репозитория программного продукта.



## 1.4.Разработка серверной части интернет-ресурса

Как уже упоминалось выше, бэкенд реализован на фреймворке Spring Boot, реализующим архитектуру REST. Предусмотрены контроллеры, сущности, репозитории. Далее будут представлены примеры, в частности представленная на рисунке 8 сущность «Пользователь» с использованием аннотаций из библиотеки Lombok.

```
@Data
@Document
public class User {

    @Id
    private String id;
    private String login;
    private String password;
    private Set<Role> roles;
}
```

Рисунок 8 – Скриншот кода класса модели пользователя

На рисунке 9 представлен класс сервиса пользователя, позволяющий сохранять новых пользователей, осуществлять поиск по уникальному идентификатору, логину и поиск всех имеющихся в хранилище данных пользователей.

```
@Service
@Slf4j
public class UserServiceImpl implements UserService {

    private final UserRepository userRepository;

    public UserServiceImpl(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public void save(User user) {
        if (userRepository.findByLogin(user.getLogin()).isEmpty()) {
            userRepository.save(user);
        }
    }

    @Override
    public User findById(String id) {
        return userRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("No such user"));
    }

    @Override
    public List<User> findAll() {
        return userRepository.findAll();
    }

    @Override
    public User findByLogin(String login) {
        return userRepository.findByLogin(login)
            .orElseThrow(() -> new RuntimeException("No such user"));
    }
}
```

Рисунок 9 – Скриншот кода сервиса пользователя

Вся конфигурация может быть удобно развёрнута благодаря настройке Docker контейнера, представленного на рисунке 10.

```
FROM gradle:6.8-jdk11

COPY ./ ./
RUN gradle clean bootJar

RUN mkdir app

ENTRYPOINT ["java","-jar", "build/libs/api-0.0.1-SNAPSHOT.jar"]
```

Рисунок 10 – Скриншот содержимого файла сборки dockerfile

### **Выводы к четвертой главе**

По прохождении четвертой главы исследовательской работы были изложены основные сведения бэкенд составляющей приложения, представлены важные и полезные для обзора листинги и представлен код сборщика проекта, продемонстрированы инструменты для взаимодействия модулей сервера между собой и между сервером и пользователем.

## 1.5.Разработка клиентской части интернет-ресурса

Для защиты данных от несанкционированного доступа других пользователей, предусмотрена система регистрации и авторизации, продемонстрированная на рисунке 11.

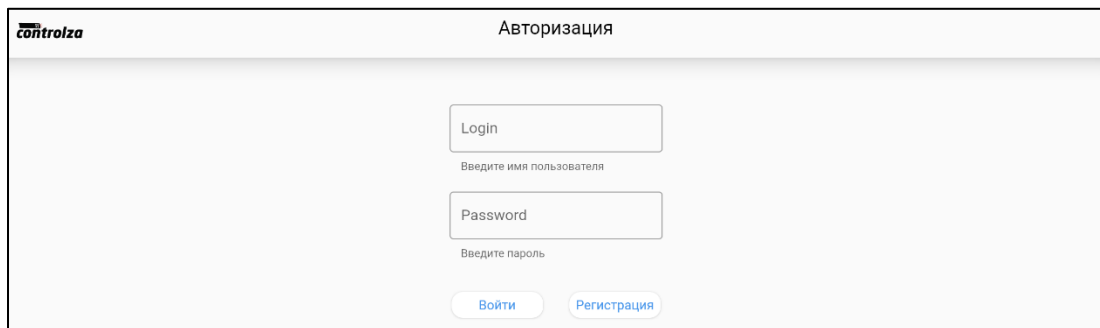


Рисунок 11 – Скриншот страницы авторизации

Поскольку в системе хранятся файлы различных версий, нужно иметь возможность выбирать версию из имеющихся – с этим справляется возможность выбора версии из выпадающего меню на странице выбора версии для скачивания, представленного на рисунке 12.

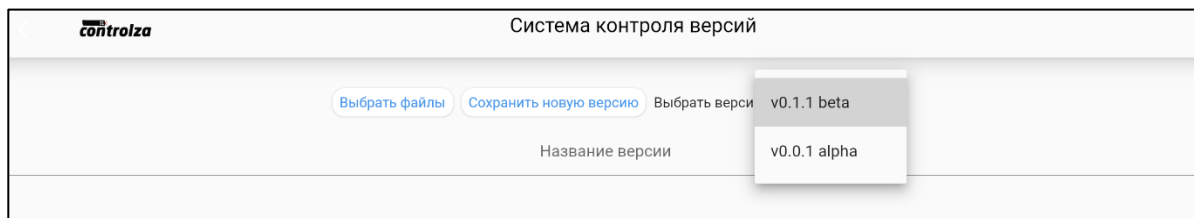


Рисунок 12 – Скриншот страницы выбора версии загруженного файла

Из представленного на рисунке 13 видно, что можно загрузить новые файлы, выбрать у них название для новой версии и сохранить файл с выбранными параметрами в систему.

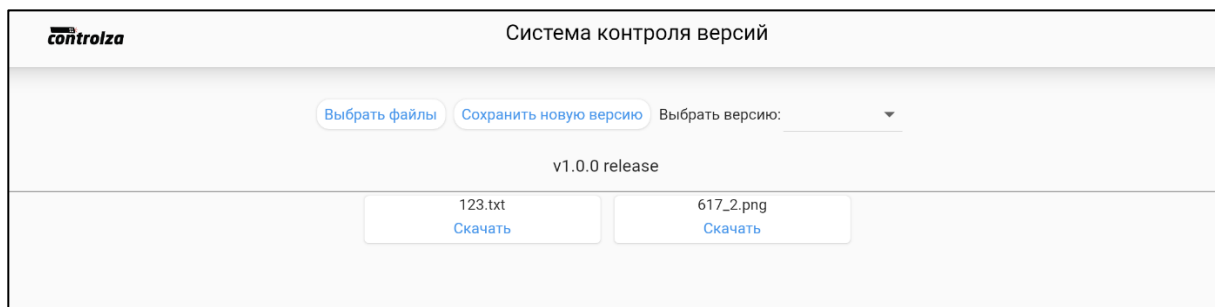


Рисунок 13 – Скриншот страницы загрузки новых файлов, выбора имени и сохранения версии

### Выводы к пятой главе

В данной главе были продемонстрированы визуальная составляющая разработанного веб-ресурса, показан и описан имеющийся функционал, предоставленный пользователю, страница регистрации и авторизации.

## 1.6. Тестирование

Хорошим тоном при разработке программных продуктов является тестирование приложений перед их эксплуатацией пользователями, поэтому крайне важно проверить основные элементы, с которыми могут взаимодействовать специалисты, использующие разрабатываемое веб-приложение. Для проверки успешной работы API был использован Postman, позволяющий проверить отправку и получение запросов между серверной и клиентской частями ресурса. На нижеследующем рисунке 14 представлена проверка запроса на регистрацию нового пользователя в системе.

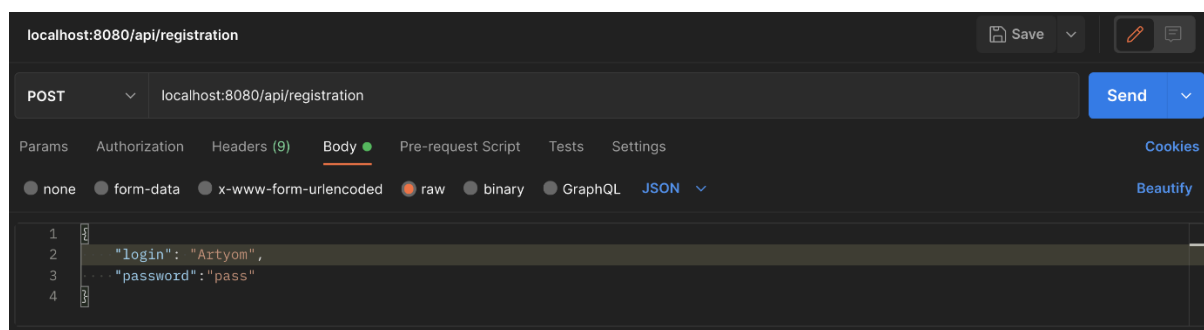


Рисунок 14 – Скриншот проверки запроса на регистрацию нового пользователя

На рисунке 15 происходит проверка пользователя на наличие в базе: если пользователь существует, то возвращается код успешного ответа, в противном случае возвращается ошибка с соответствующим кодом.

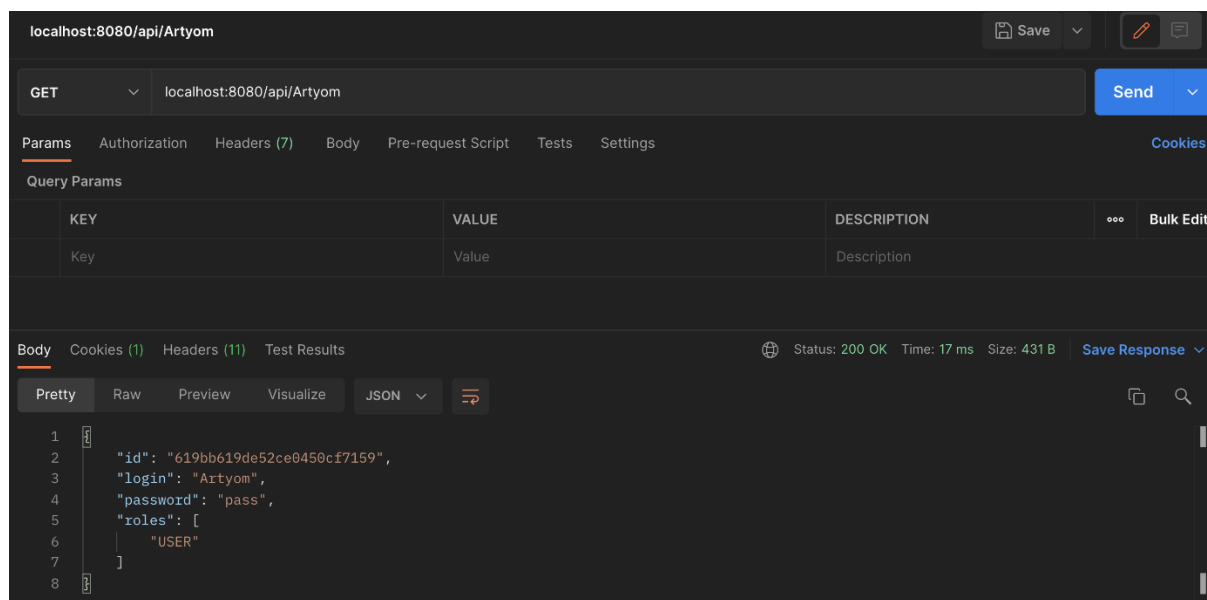


Рисунок 15 – Скриншот проверки запроса авторизации пользователя

Чтобы проверить наличие пользователей в базе данных с целью тестирования их успешной регистрации, используется GET запрос */all*, что можно увидеть на рисунке 16.

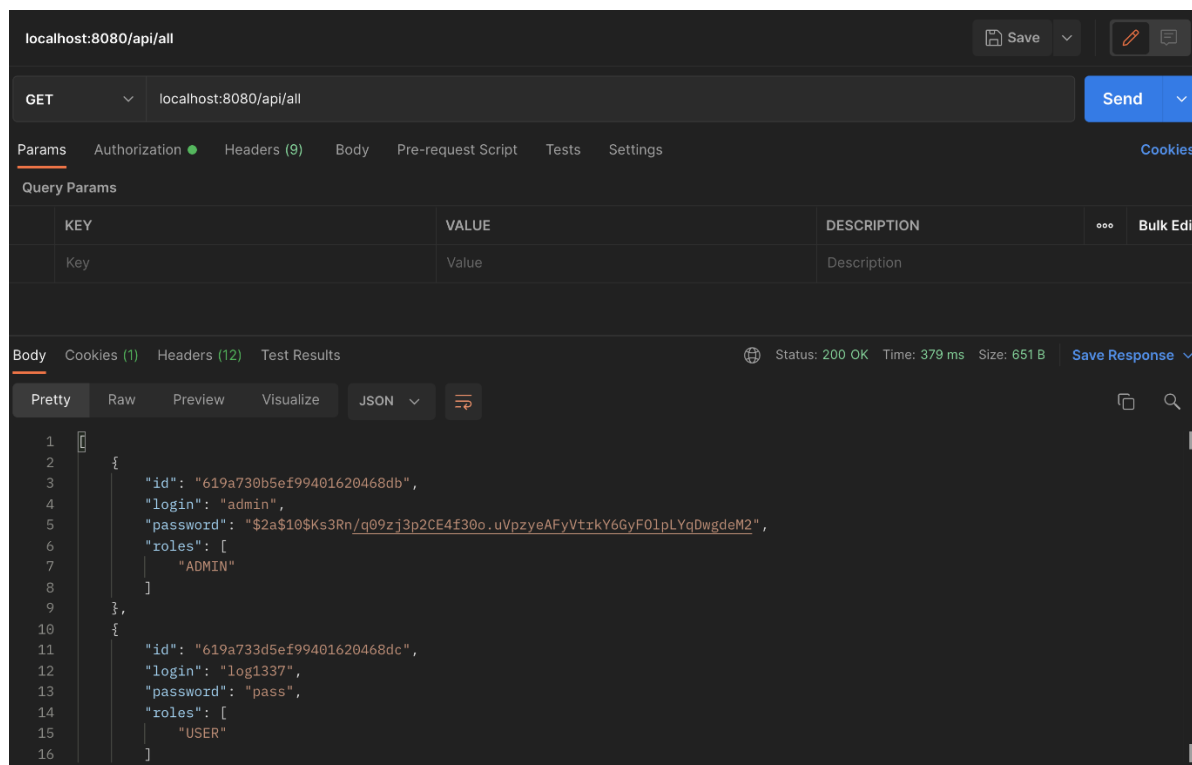


Рисунок 16 – Скриншот проверки запроса на получение списка пользователей

## Выводы к шестой главе

Web-приложение успешно прошло все представленные тесты, в результате чего можно сделать вывод о надежной и верной работе клиентской и серверной частей веб-приложения, возможности использования данного приложения реальными пользователями.



## **Заключение**

По завершении выполнения данной курсовой работы было проведено исследование механизма работы версионного контроля, разработан собственный прототип, удовлетворяющий все поставленные функциональные требования, получен ценный опыт разработки веб-приложений и программ в целом, закреплены знания, полученные на дисциплинах «Архитектуры клиент-серверных приложений», «Разработка серверных частей интернет ресурсов» и «Интерфейсы прикладного программирования» данного учебного семестра.

## 2. Список литературы

1. Майкл, С. М. Разработка одностраничных веб-приложений / С. М. Майкл, К. П. Джош ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2014. — 512 с. — ISBN 978-5-97060-072-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/69951> (дата обращения: 16.11.2021). — Режим доступа: для авториз. пользователей.
2. Уоллс, К. Spring в действии / К. Уоллс. — Москва : ДМК Пресс, 2013. — 752 с. — ISBN 978-5-94074-568-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/39988> (дата обращения: 16.11.2021). — Режим доступа: для авториз. пользователей.
3. Соснин, П. И. Архитектурное моделирование автоматизированных систем : учебник / П. И. Соснин. — Санкт-Петербург : Лань, 2020. — 180 с. — ISBN 978-5-8114-3919-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/130183> (дата обращения: 16.11.2021). — Режим доступа: для авториз. пользователей.
4. Тузовский, А. Ф. Проектирование и разработка web-приложений : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2021. — 218 с. — (Высшее образование). — ISBN 978-5-534-00515-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/469982> (дата обращения: 16.11.2021).
5. PHP 7 / Д.В. Котеров, И. В. Симдянов. - СПб.: БХВ-Петербург, 2021. - 1088 с.: ил. (дата обращения: 16.11.2021).
6. Хоффман Эндрю X85 Безопасность веб-приложений. — СПб.: (дата обращения: 16.11.2021).
7. Питер, 2021. — 336 с.: ил. — (Серия «Бестселлеры O'Reilly») (дата

- обращения: 16.11.2021).
8. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — СПб. : Питер, 2021. — 352 с. (дата обращения: 16.11.2021).
  9. Персиваль, Г. Паттерны разработки на Python: TDD, DDD и событийно-ориентированная архитектура / Г. Персиваль, Б. Грегори. — СПб. : Питер, 2022. — 336 с. (дата обращения: 16.11.2021).
  10. Раджпут Д. Spring. Все паттерны проектирования. - СПб.: Питер, 2019. (дата обращения: 16.11.2021).
  11. Меджуи М., Уайлд Э., Митра Р., Амундсен М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. - СПб.: Питер, 2020. (дата обращения: 16.11.2021).
  12. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 5-е изд.. - СПб.: Питер, 2021. (дата обращения: 16.11.2021).
  13. Бэнкс А., Порселло Е. GraphQL: язык запросов для современных веб-приложений. - СПб.: Питер, 2019. (дата обращения: 16.11.2021).
  14. Антонова И. И., Кашкин Е. В. Разработка web-сервисов с использованием HTML, CSS, PHP и MySQL [Электронный ресурс]: учебно -методическое пособие. - М.: РТУ МИРЭА, 2019. - — Режим доступа: <http://library.mirea.ru/secret/15052019/2022.iso> (дата обращения: 16.11.2021).
  15. Диков А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3 [Электронный ресурс]: учебное пособие. - Санкт-Петербург: Лань, 2019. - 188 с. — Режим доступа: <https://e.lanbook.com/book/122174> (дата обращения: 16.11.2021).