DOI 10.37882/2223–2966.2022.12–2.26

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «ТЕХКООПЕРАЦИЯ 2.0»

DEVELOPMENT OF THE INFORMATION SYSTEM "TECHNICAL COOPERATION2.0"

M. Polenok
S. Bondarenko
O. Zelenskiy
O. Yurkova

Summary. The information system "Technical Cooperation 2.0" has been developed, which has a convenient and concise design. It is able to store and process information about the enterprises of registered users, as well as their offers or requests. The analysis of the theoretical aspects of the implementation of b2b systems was carried out, the key stages of the development of such systems were identified and analyzed in detail. As a result of the analysis of existing software solutions, the disadvantages of existing systems were identified and the elements that can distinguish the system being created from all others were emphasized.

Keywords: information system, b2b system, Java< Spring Boot.

Поленок Максим Викторович
Брянский государственный инженерно-технологический университет
polenok.maksim.2001@mail.ru
Бондаренко Сергей Владимирович
Брянский государственный инженерно-технологический университет
Bondrenkoseregabondarenko576@gmail.com
Зеленский Олег Сергеевич
Брянский государственный инженерно-технологический университет
zelenskiyoleg2000@gmail.com
Юркова Ольга Николаевна
К.э.н., Брянский государственный инженерно-технологический университет
yurkova_olga@mail.ru

Аннотация. Разработана информационная система «Техкооперация 2.0», которая имеет удобный и лаконичный дизайн. Она способна хранить и обрабатывать информацию о предприятиях зарегистрированных пользователей, а также их предложениях или запросах. Произведен анализ теоретических аспектов реализации b2b систем, были выявлены и подробно разобраны ключевые этапы разработки подобных систем. В результате анализа существующих программных решений были выявлены минусы существующих систем и подчеркнуты элементы, способные выделить создаваемую системы из всех остальных.

Ключевые слова: информационная система, b2b система, Java < Spring Boot.

Введение

Актуальность разрабатываемой системы заключается в том, что в настоящее время в стране вопрос импортозамещения является одним из ключевых. Сотрудничество с иностранными предприятиями становится все более трудоёмким и несет все больше материальных затрат.

Для решения данной проблемы было решено разработать информационную систему «Техкооперация 2.0». Данная интернет система представляет из себя частный случай b2b систем. Она позволяет предпринимателю найти предприятие способное производить интересующую его продукцию и связаться с ним, а также предложить свои производственные показатели.

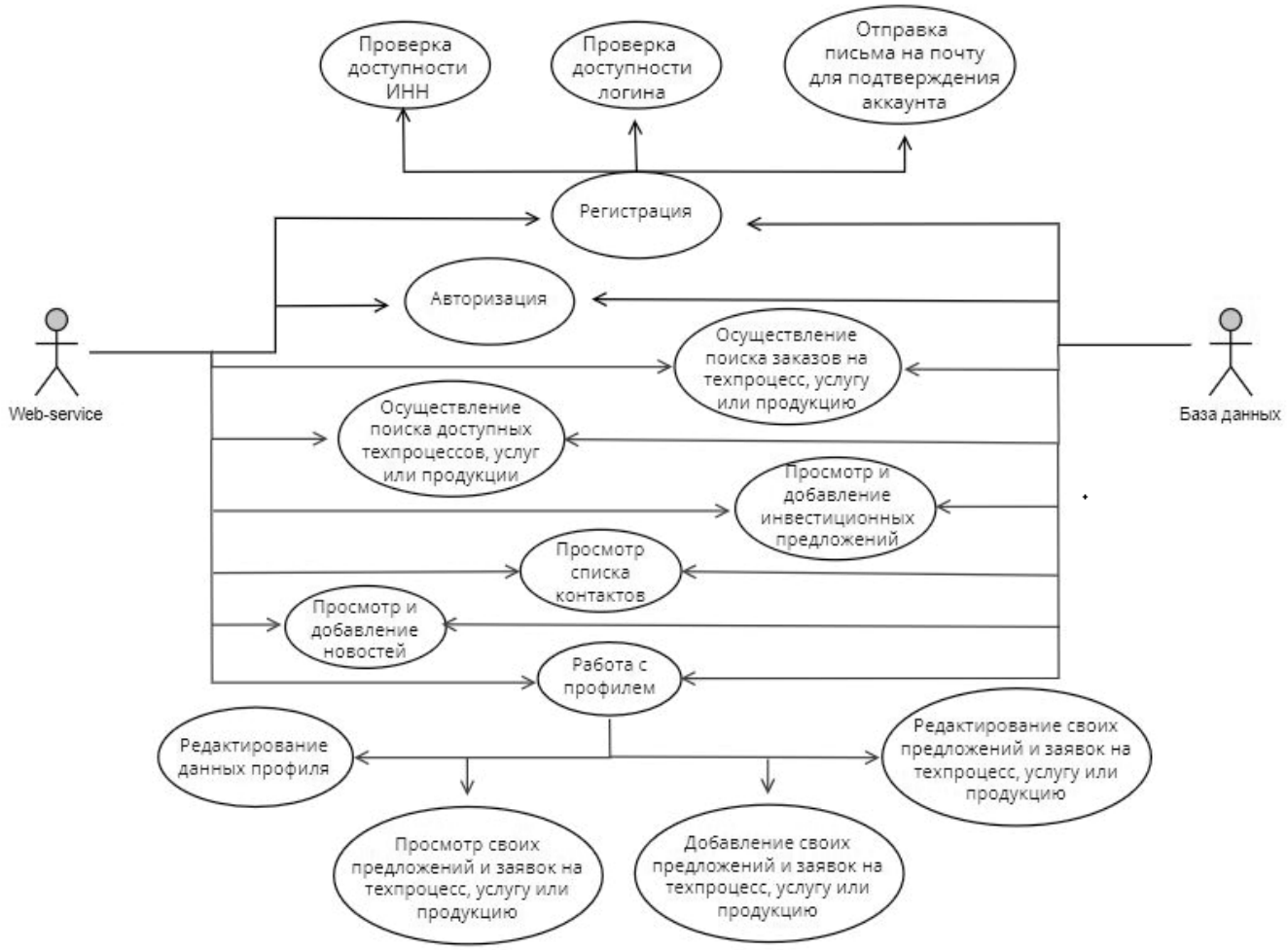


Рис. 1. Диаграмма вариантов использования информационной системы «Техкооперация 2.0»

Объектом исследования является информационная система «Техкооперация 2.0».

Предмет исследования: бэкенд-сервис информационной системы «Техкооперация 2.0».

Цель работы состоит в разработке бэкенд-сервиса информационной системы «Техкооперация 2.0».

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать теоретические аспекты реализации b2b систем;
- произвести анализ существующих программных решений;
- определить спецификации и произвести проектирование бэкенд сервиса информационной системы «Техкооперация 2.0»;
- реализовать бэкенд-сервис информационной системы «Техкооперация 2.0»;
- произвести тестирование системы.

Теоретические аспекты и основные этапы создания b2b систем

B2B система — это система, способствующая организации торговли в среде оптовиков, то есть крупных компаний. При этом одна и так же компания может выступать в системе как закупщик и как поставщик товаров или услуг.

Одной из ключевых целей создания b2b систем является: помощь предприятиям в налаживании бизнес-схем и связей, что в дальнейшем способствует увеличению прибыли.

Чтобы создать хорошую b2b систему, необходимо учесть множество деталей. Весь процесс создания b2b системы можно разделить на 3 крупных шага: под-

Рис. 2. Диаграмма последовательности процесса регистрации пользователя в системе

готовка плана разработки, разработка функционала системы и третий этап завершение и запуск системы. Рассмотрим подробно каждый из этапов создания системы.

Первый этап — это подготовка плана разработки. На данном этапе происходит определение целей систем, описание ожидаемого функционала с учетом поведения пользователей.

На основе целей и ожидаемого функционала составляется техническое задание. В данном документе четко определяются цели, задачи и предпосылки к развитию проекта в будущем.

Не малую роль в данном этапе играет разработка дизайна системы. Может показаться, что дизайн не имеет отношения к автоматизации бизнес-процессов, но именно благодаря красоте и удобству пользования удается привлечь и удержать большую часть клиентов системы.

Второй этап — это разработка функционала системы. На данном этапе происходит создание основной части проекта. Проектируется и реализуется серверная часть «back-end», в которой реализуются функции, отвечающие за хранение и манипулировании информацией.

На основе разработанного дизайна создается клиентская часть «front-end», которая отображается данные, хранящиеся в серверной части, а также предоставляет удобный функционал пользовательского интерфейса для манипулирования данными. Также на данном этапе производится тестирование готовой системы и исправление возникающих ошибок.

Третий этап — это завершение и запуск системы. В данном этапе производится наполнение системы первичным контентом, выгрузка системы на конечные сервера, тестирование отказоустойчивости и исправление проблем.

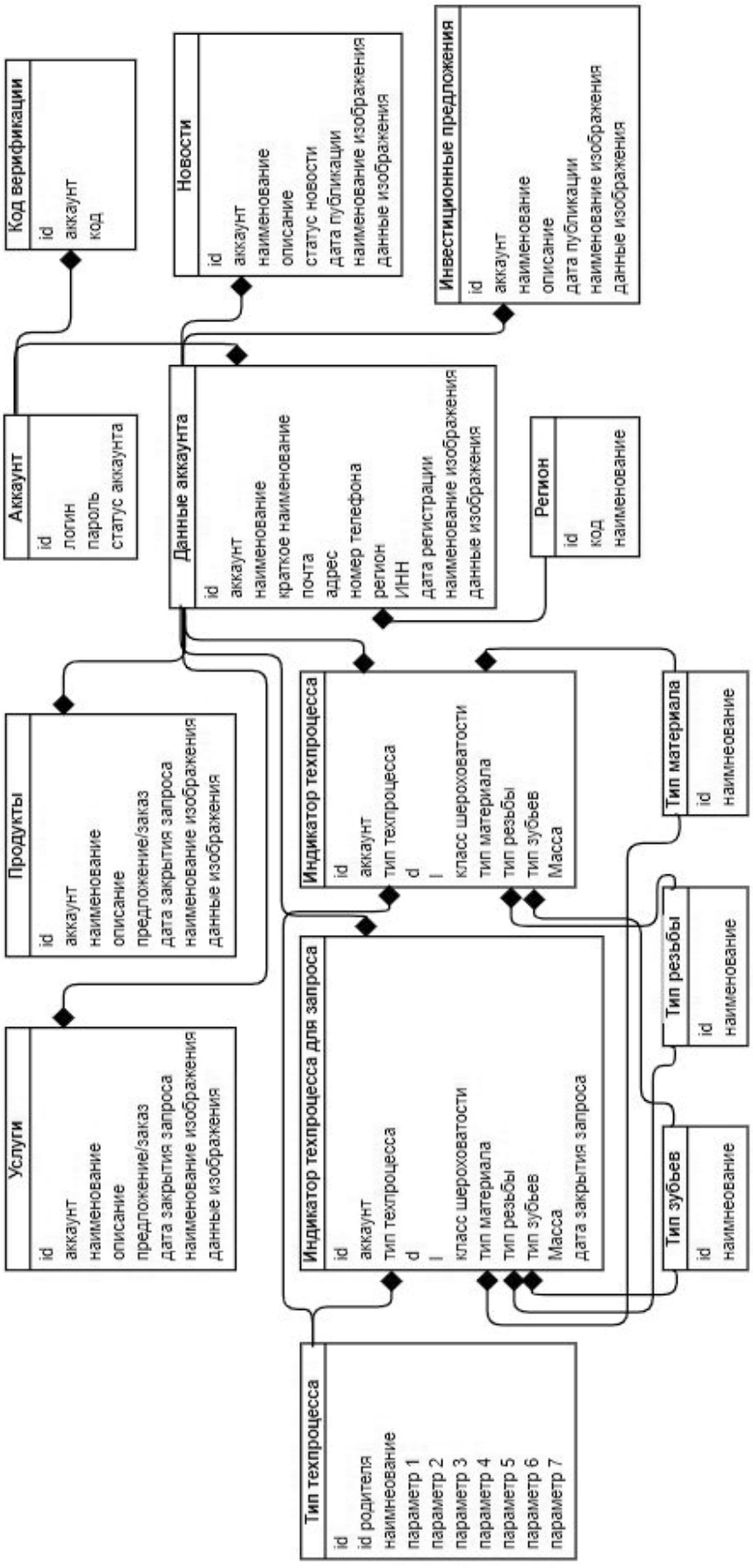


Рис. 3. Диаграмма классов информационной системы «Техкооперация 2.0»

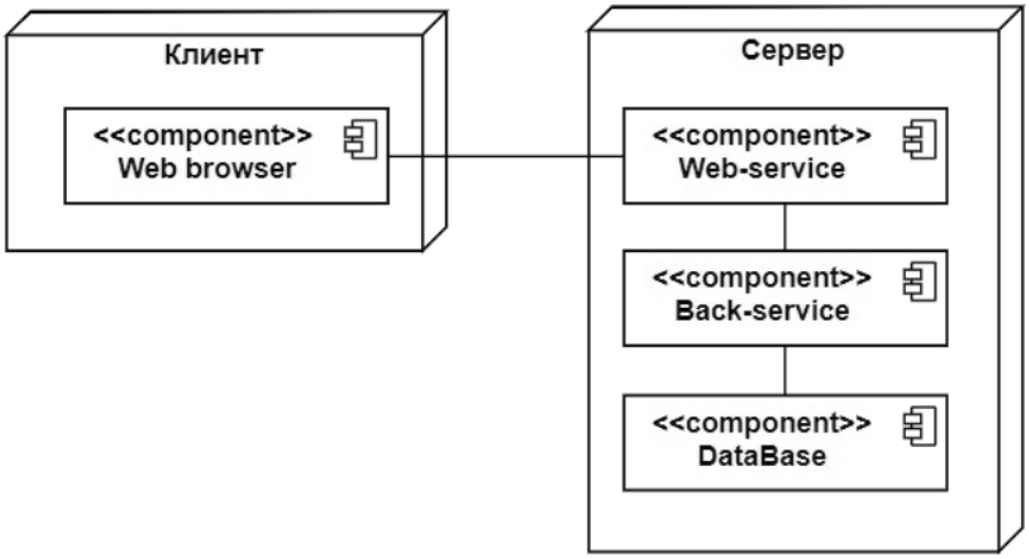


Рис. 4. Диаграмма развертывания

После всевозможных тестов осуществляется запуск системы в общее пользование.

Определение спецификаций и проектирование информационной системы «Техкооперация 2.0»

При разработке любой системы необходимо чётко понимать: для чего разрабатывается система, и какие процессы будут протекать в системе. Для этого в ходе выполнения работы была разработана диаграмма вариантов использования (рис. 1).

Диаграмма вариантов использования отражает функциональные возможности системы, используя действующие лица, актеры. На рисунке 1 действующими лицами выступают «Web-service» и «База данных».

«Web-service» взаимодействует с базой данных при помощи следующих вариантов использования:

- регистрация аккаунта пользователя;
- авторизация пользователя;
- просмотр и добавление новостей;
- просмотр и добавление инвестиционных предложений;
- просмотр списка доступных техпроцессов, услуг и продуктов;
- просмотр списка заказов на техпроцесс, услугу и продукцию;
- возможность через профиль осуществлять добавление и редактирование предложения или заказа на техпроцесс, услугу продукцию;
- редактирование профиля.

Для того чтобы детально разобраться в том, как взаимодействуют между собой разные элементы системы, необходимо разработать диаграмму последовательностей (рис. 2).

Основными элементами диаграммы являются актеры, которые изображаются в виде прямоугольников, и вертикальные линии, называемые линиями жизни процесса.

При разработке любой информационной системы очень важно определить основные компоненты системы. Одним из самых удобных способов отражения функциональных компонентов системы является диаграмма классов.

Диаграмма классов — это структурная диаграмма языка моделирования UML, на которой отражаются основные объекты системы, статические связи между ними, а также атрибуты классов (рис. 3).

На рисунке 4 можно заметить, что все основные элементы системы располагаются на сервере. Для обращения к системе клиенту достаточно иметь любой браузер. Также можно заметить, что клиент обращается только к «Web-service», который взаимодействует со всеми остальными элементами информационной системы.

Реализация функциональной модели информационной модели «Техкооперация 2.0»

Рассмотрим основные компоненты Back-сервиса информационной системы «Техкооперация 2.0». Начнем разбор с ключевых файлов:

```
server.port=5534

spring.datasource.url=jdbc:mysql://localhost:3306/baz?serverTimezone=UTC
spring.datasource.username=JavaApp
spring.datasource.password=sql2

spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.properties.hibernate.current_session_context_class=org.springframework.orm.hibernate5.SpringSessionContext

spring.mail.host=smtp.yandex.ru
spring.mail.username=tehprocess@bgitu.ru
spring.mail.password=*******
spring.mail.port=465
spring.mail.protocol=smtps
mail.debug=false
```

Рис. 5. Файл настроек SpringBoot

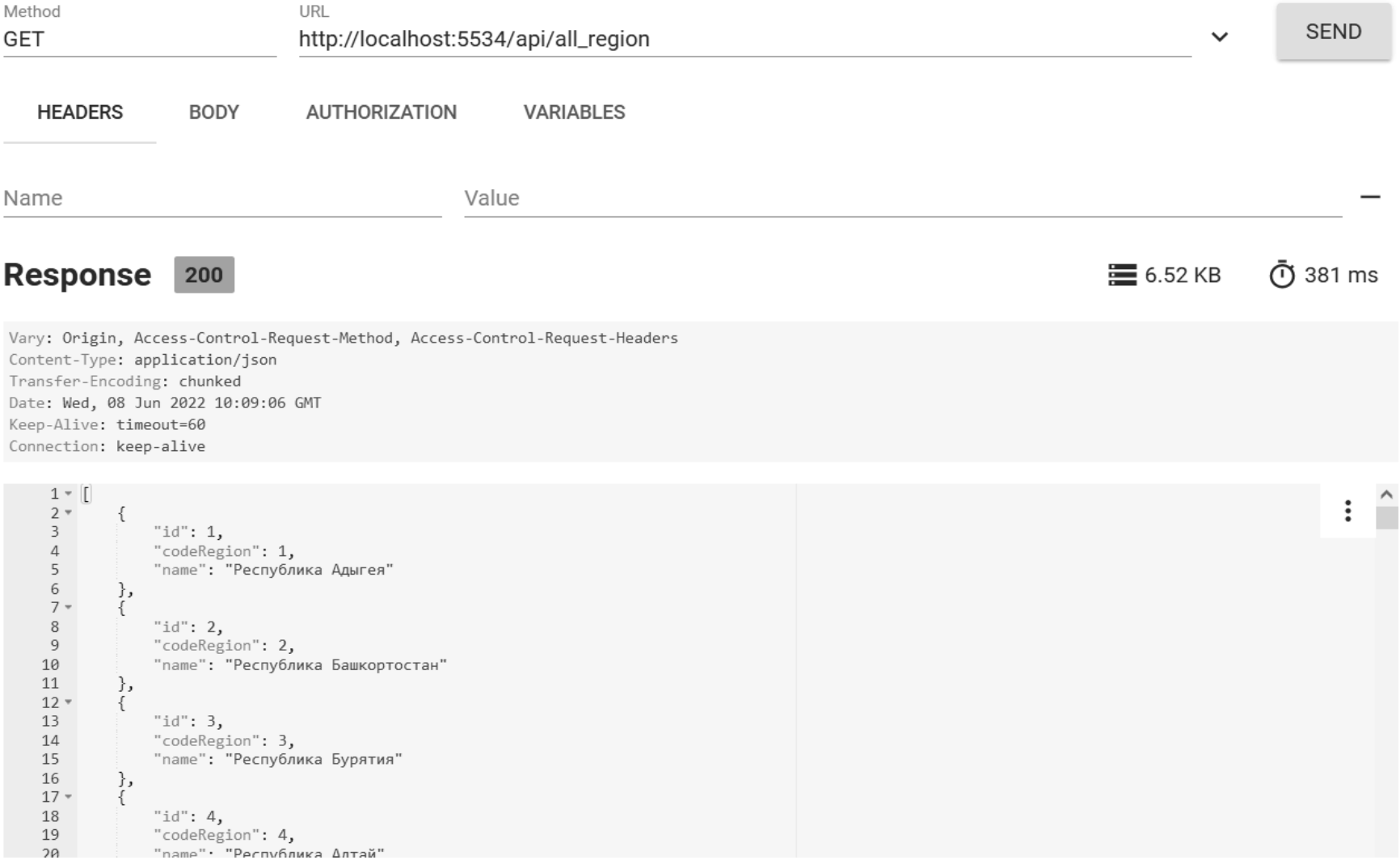


Рис. 6. Пример работы метода getAllRegion

- DataBaseServiceApplication.java;
- Pom.xml;
- Application.properties.

Класс «DataBaseServiceApplication.java» является точкой старта системы. Данный класс содержит код, отраженный в листинге 1.

Листинг 1 — Файл DataBaseServiceApplication.java

```
@SpringBootApplication
public class DataBaseServiceApplication {
public static void main(String[] args) {
SpringApplication.run(DataBaseServiceApplication.
class, args);
```

Рассматриваемый класс содержит метод «main» и аннотацию «@SpringBootApplication», которая сообщает фреймворку «SpringBoot», с какого места следует начинать запуск программы.

Файл «pom.xml» содержит список подключаемых к проекту зависимостей и их версии. Ознакомиться с подробным списком зависимостей можно в листинге 2.

Листинг 2 — Подключаемые зависимости приложения

```
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.apache.tomcat.embed</groupId>
<artifactId>tomcat-embed-jasper</artifactId>
<version>9.0.44</version>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
<version>2.6.2</version>
</dependency>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.21</version>
</dependency>
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-core</artifactId>
<version>5.6.3.Final</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-mail</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
```

Рассмотрим некоторые из зависимостей.

Зависимость «spring-boot-starter-web» указывает, с какой из подсистем фреймворка SpringBoot мы будем работать.

Зависимость «mysql-connector-java» указывает СУБД, в которой будет располагаться база данных и подключает необходимый для работы с ней плагин.

Зависимости «spring-boot-starter-data-jpa» и «hibernate-core» подключают необходимые библиотеки для работы с базой данных.

Зависимость «tomcat-embed-jasper» подключает контейнер сервлетов, необходимый для работы веб-приложений.

Зависимость «spring-boot-starter-mail» подключает библиотеки необходимые для отправки писем по электронной почте из java-приложения.

Файл «application.properties» содержит настройки для SpringBoot (рис. 5).

Рассмотрим пакет «config». Данный «пакет» содержит всего один класс конфигурации, «MailConfig». Ознакомиться с кодом класс можно в листинге 3.

Листинг 3 — Файл «MailConfig»

```
@Configuration
public class MailConfig {
@Value("${spring.mail.host}")
private String host;
@Value("${spring.mail.username}")
private String username;
@Value("${spring.mail.password}")
private String password;
@Value("${spring.mail.port}")
```

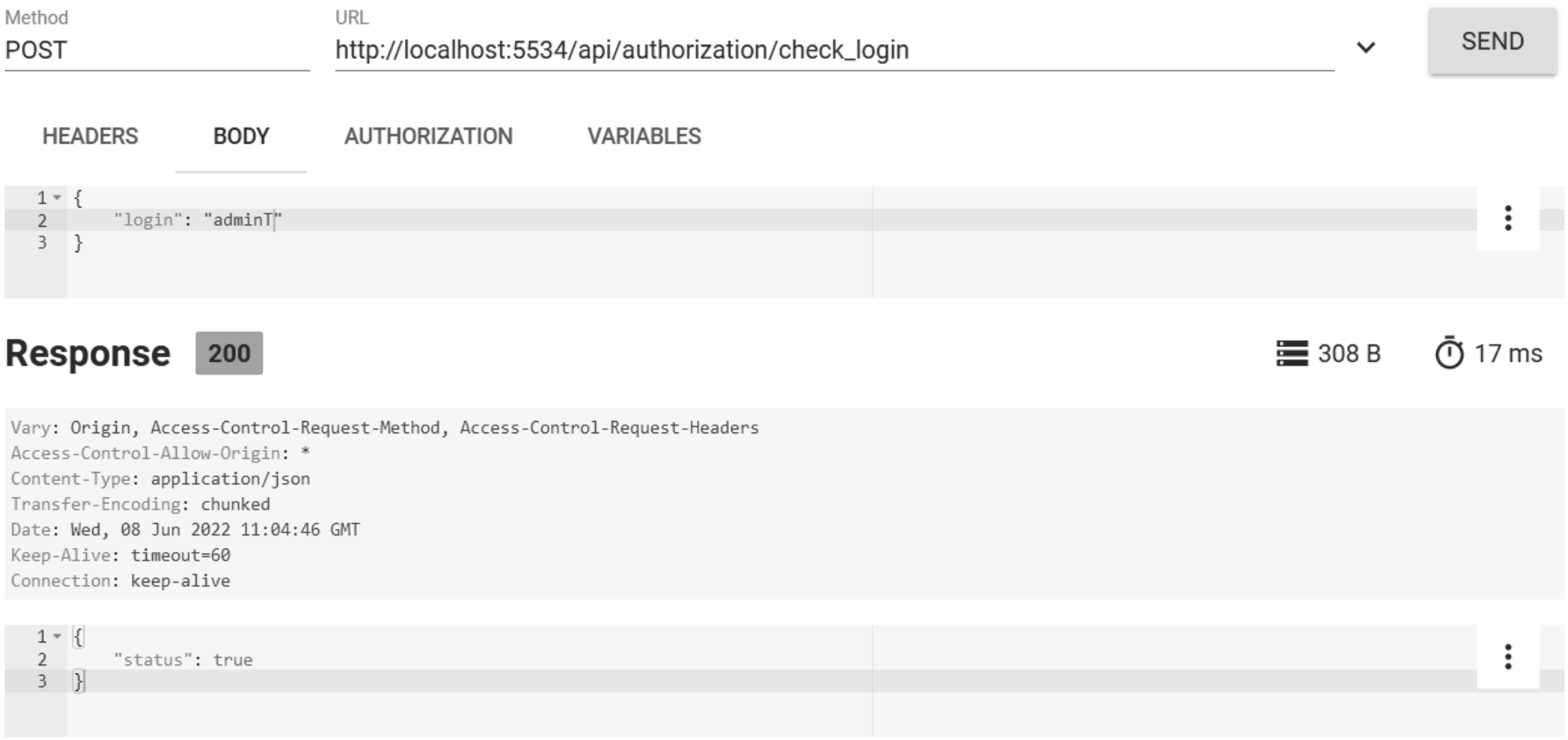


Рис. 7. Пример работы метода getCheckLogin

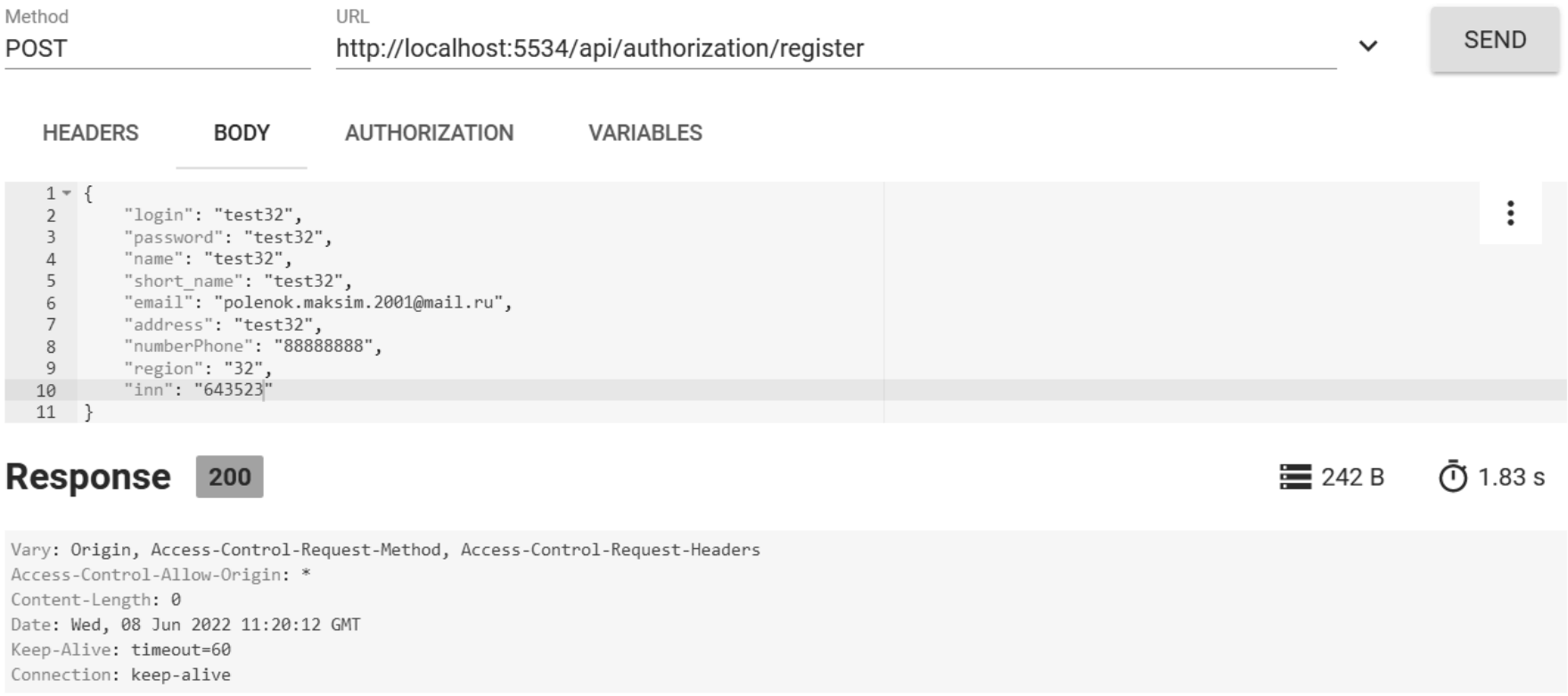


Рис. 8. Пример работы метода getRegistration

```
    private int port;
    @Value("${spring.mail.protocol}")
    private String protocol;
    @Value("${mail.debug}")
    private String debug;
    @Bean
    public JavaMailSender getMailSender() {
    JavaMailSenderImpl mailSender = new
JavaMailSenderImpl();
    mailSender.setHost(host);
    mailSender.setPort(port);
    mailSender.setUsername(username);
    mailSender.setPassword(password);
    Properties properties = mailSender.
getJavaMailProperties();
    properties.setProperty("mail.transport.protocol",
protocol);
    properties.setProperty("mail.debug", debug);
    return mailSender;
    }
    }
```

Данный класс необходим для создания и настройки переменной, отвечающей за отправку сообщений по электронной почте.

Следующим рассмотрим каталог service. В данном каталоге содержатся классы, содержимое которых неоднократно используется в методах других функциональных классов.

В данном каталоге содержится один класс, «MailSender». В листинге 4 можно подробнее ознакомиться с содержимым класса.

Листинг 4 — Файл «MailSender»

```
    @Service
    public class MailSender {
    @Autowired
    private JavaMailSender mailSender;
    @Value("${spring.mail.username}")
    private String username;
    public void send(String emailTo, String subject, String
message) {
    SimpleMailMessage mailMessage = new
SimpleMailMessage();
    mailMessage.setFrom(username);
    mailMessage.setTo(emailTo);
    mailMessage.setSubject(subject);
    mailMessage.setText(message);
    mailSender.send(mailMessage);
    }
    }
```

Данный класс содержит функции формирования и отправки сообщения по электронной почте.

Тестирование и апробация программного средства

Рассмотрим пример работы основных методов Backservice.

Метод getAllRegion возвращает список всех регионов в формате JSON. Для того чтобы обратиться к данному методу из интернета, необходимо отправить GETREST-запрос по адресу: http://localhost:5534/api/all_region (рис. 6).

Методы getAllMarerial, getAllCarvingType, getAllToothingType работают по идентичному принципу.

Метод getRegistration принимает данные нового аккаунта в формате JSON и регистрирует его в системе с пометкой «не активирован». На рис. 8 можно ознакомиться с примером работы метода.

Методы getCheckLogin и getCheckInn идентичны, поэтому рассмотрим один из них. Метод getCheckLogin вызывается при попытке регистрации нового аккаунта. Он принимает логин в формате JSON и проверяет его наличие в базе данных. Если данный логин присутствует, то возвращается «true» иначе «false». Для того чтобы обратиться к данному методу из интернета, необходимо отправитьPOSTREST-запрос по адресу: http://localhost:5534/api/authorization/check_login (рис. 7).

При работе метода, описанного выше, мы не сможем сразу зайти в зарегистрированный аккаунт. По окончанию регистрации метод отправляет электронное письмо для авторизации аккаунта на указанную почту при регистрации. Для активации аккаунта необходимо перейти по ссылке в письме.

Заключение

Итогом работы служит достижение цели, т.к. была разработана информационная система «Техкооперация 2.0», которая имеет удобный и лаконичный дизайн. Данная система способна хранить и обрабатывать информацию о предприятиях зарегистрированных пользователей, а также их предложениях или запросах.

После подробного анализа теоретических аспектов реализации b2b систем были выявлены и подробно разобраны ключевые этапы разработки подобных систем. Был сделан вывод о том, что в настоящее время про-

граммы, которые помогают развиваться бизнесу, имеют большое значение и очень востребованы на рынке.

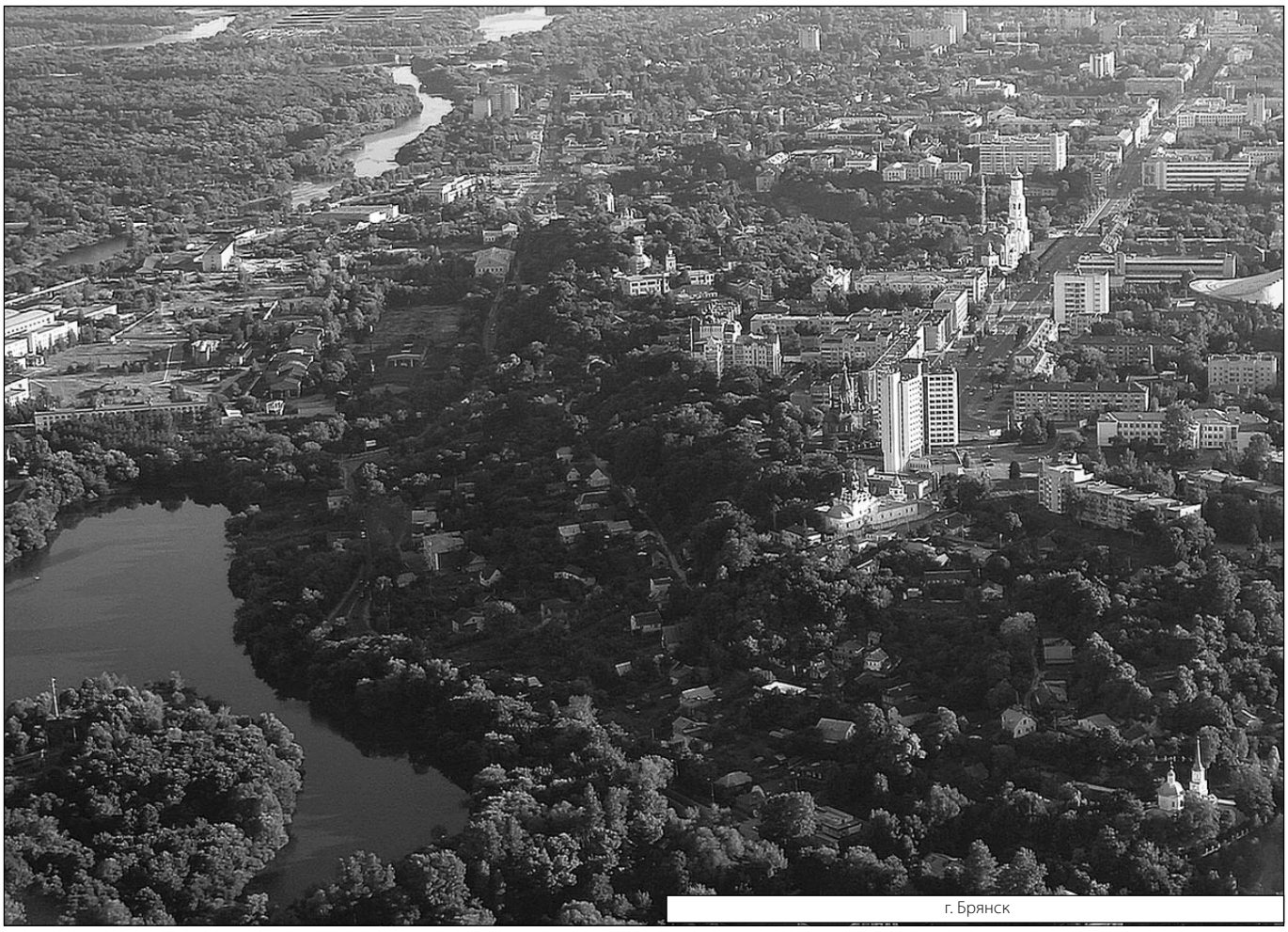
В результате анализа существующих программных решений были выявлены минусы существующих систем и подчеркнуты элементы, способные выделить создаваемую системы из всех остальных. Большинство существующих систем созданы для крупных предприятий и профессионалов торговли. Новому пользователю не всегда просто разобраться во всех протекающих процессах.

Для реализации Back-service информационной системы «Техкооперация 2.0» было решено использовать язык программирования Java, причем ключевым инструментом разработки стал фреймворк SpringBoot.

ЛИТЕРАТУРА

1. Васильев А.Н. Программирование на Java для начинающих.— Москва: Эксмо, 2020.— 704 с.
2. Дашнер Себастьян Изучаем JavaEE. Современное программирование для больших предприятий — СПб.: Питер, 2018.— 384 с.
3. Дюбуа Поль MySQL. Сборник рецептов — Символ-Плюс, 2004.— 1056 с.
4. Карнелл Дж., Санчес И.У. Микросервисы Springв действии — ДМК Пресс, 2021.— 490 с.
5. Лафоре Р. Структуры данных и алгоритмы в Java. Классика ComputersScience. 2-у изд.— СПб.: Питер, 2013.— 704с.
6. Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста.— СПб.: Питер, 2013.— 464 с.
7. Раджпут Динеш Spring. Все патерны проектирования — СПб.: Питер, 2019.— 320 с.

Журнал «Современная наука: актуальные проблемы теории и практики»



г. Брянск