МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. Шухова»

(БГТУ им. В. Г. Шухова)

**КУРСОВАЯ РАБОТА**

по дисциплине «Интерфейсы ВС»

предметная область: Ведение учета клиентов

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Обенг К.Б.

(подпись) ВТ-42

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. пр.

(подпись) Торопчин Д.А.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Белгород

2021 г.

Оглавление

[Введение 3](#_Toc30747040)

[Глава 1. Теория 4](#_Toc30747041)

[1.1. Технология Java и каково ее применения 4](#_Toc30747042)

[1.2. Модель клиент-сервер 4](#_Toc30747043)

[1.3. Spring Framework 6](#_Toc30747044)

[1.4. Angular 7](#_Toc30747045)

[Часть 2. Практика 7](#_Toc30747046)

[2.1. Постановка задачи 7](#_Toc30747047)

[2.2. Описание предметной области 8](#_Toc30747048)

[2.3. Проектирование информационной системы 8](#_Toc30747049)

[2.4. Разработка приложения 9](#_Toc30747050)

[Spring Boot Server 9](#_Toc30747051)

[Заключение 10](#_Toc30747052)

[Результаты выполнения работы 11](#_Toc30747053)

[Список литературы 13](#_Toc30747054)

[основной программный код (Spring Boot Server) 14](#_Toc30747055)

[Приложение А – Dependency 14](#_Toc30747056)

[Приложение Б- Клиент – модель 15](#_Toc30747057)

[Приложение В - Конфигурация для источника данных Spring и свойств JPA 16](#_Toc30747058)

[Приложение Г - REST Controller 16](#_Toc30747059)

# **Введение**

Возможность передачи информации на расстоянии интересовала людей всегда. Для того чтобы обмениваться информацией не только при личной встрече, но и на больших расстояниях, люди изобретали всё новые технические средства, протягивали структурированные кабельные сети по всему миру, организовывали почтовые системы, запускали спутники связи. С развитием информационных технологий стали возможным еще более глобальные коммуникации. Сегодня люди с легкостью по всему миру могут обмениваться не только сообщениями, но и графикой и звуковой и другими видами информации. Историческим «докомпьютерным» предшественником чатов, несомненно, являлся телефон. Ни телеграф, ни почта не могли позволить общаться в реальном времени и не были доступны в домашних условиях. Изобретение и распространение телефона вызвало настоящую революцию способах и средствах общения. Возможность поговорить с собеседником в реальном времени на другом континенте казалась настоящим чудом. Язык Java является основой практически для всевозможных типов сетевых приложений и всеобщим стандартом для разработки встроенных и мобильных приложений, веб-контента, игр, и корпоративного программного обеспечения. В мире насчитывается более девяти миллионов специалистов, разрабатывающих приложения на Java, которая позволяет эффективно разрабатывать, тестировать, внедрять и использовать программное обеспечение.

# **Глава 1. Теория**

## **Технология Java и каково ее применения**

Java представляет собой язык программирования и платформу вычислений, которая была впервые выпущена Sun Microsystems в 1995 г. Существует множество приложений и веб-сайтов, которые не работают при отсутствии установленной Java, и с каждым днем число таких веб-сайтов и приложений увеличивается. Java отличается быстротой, высоким уровнем защиты и надежностью. От портативных компьютеров до центров данных, от игровых консолей до суперкомпьютеров, используемых для научных разработок, от сотовых телефонов до сети Интернет. После загрузки Java пользователи получают Java Runtime Environment (JRE). JRE состоит из Java Virtual Machine (JVM), базовых классов платформы Java и вспомогательных библиотек платформы Java. JRE является областью программного обеспечения Java, используемой во время выполнения, т.е. единственным компонентом, который требуется для его запуска в используемом в веб-браузере. Программное обеспечение подключаемого модуля Java является компонентом Java Runtime Environment (JRE). JRE обеспечивает возможность запуска апплетов, написанных на языке программирования Java, в различных браузерах. Подключаемый модуль Java не является автономной программой и не может быть установлен отдельно. Виртуальная машина Java представляет собой только один аспект программного обеспечения Java, который задействуется при взаимодействии через Интернет. Виртуальная машина Java встраивается непосредственно в загрузку программного обеспечения Java, и используется для поддержки запуска приложений Java.

## **Модель клиент-сервер**

Процесс разработки программного обеспечения – набор правил, согласно которой построена разработка программного обеспечения. Приложение можно назвать клиент серверным если оно включает в себя клиент-серверную архитектуру. Разработку клиент-серверного приложения необходимо начинать с выбора архитектуры клиент-сервера. Архитектура «клиент-сервер» характеризуется наличием по крайне мере двух взаимодействующих, самостоятельных процессов – клиента и сервера. Процессы, осуществляющие некоторую службу, например, службу базы данных или файловой системы, называются серверами, а процессы, запрашивающие службы у серверов посредством посылки запроса и последующего ожидания ответа от сервера, называются клиентами. Фактически эти процессы – программное обеспечение, которое установлено на разных вычислительных машинах и взаимодействующее между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине.

Diagram

Description automatically generated

Любая архитектура клиент-сервера должна включать в себя три основных компонента:

• ввод и отображение данных (интерфейс с пользователем);

• прикладные функции, характерные для данной предметной области;

• управление информационными ресурсами (базой даных или файловой системой и т.д).

В классической архитектуре клиент-сервер необходимо распределять три основные части приложения по двум физическим программным обеспечениям.

Многоуровневая архитектура (N-tier или multi-tier, иногда его называют трехуровневая архитектура или трехзвенная архитектура, но это частный случай) представляет собой дальнейшее совершенствование технологии «клиент-сервер». В трехзвенной архитектуре вся обработка данных, ранее входившая в клиентские приложения, или полностью была на сервере либо эти два звена делили её функции между собой, выделяется в отдельное звено, называемое сервером приложений.

A picture containing text, device, gauge

Description automatically generated

Плюсы многоуровневой архитектуры:

• Масштабируемость (за счёт горизонтальной масштабируемости сервера приложений и мультиплексирования соединений);

• Интегрируемость (за счёт изолированности уровней друг от друга);

• Доступность из веб-браузера или из тонкого клиента (т.к клиент занимается только отображением информации пользователю в удобном виде);

• Высокая безопасность;

Минусы:

• Высокая стоимость, дополнительные расходы на администрирование;

• Разработка такой системы гораздо сложнее чем двухзвенной. Клиент-серверная система должна иметь достойный уровень безопасности информации, хранимой и передаваемой между звеньями системы.

## **Spring Framework**

Spring Framework – универсальный фреймворк с открытым исходным кодом для Java - платформы. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода. **Spring Boot** обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов. Чтобы ускорить процесс управления зависимостями, Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых **starter**-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т.д.). **Starter**-пакеты представляют собой набор удобных дескрипторов зависимостей, которые можно включить в свое приложение. Это позволит получить универсальное решение для всех, связанных со Spring технологий, избавляя программиста от лишнего поиска примеров кода и загрузки из них требуемых дескрипторов зависимостей. Другими словами, **Spring Boot**собирает все общие зависимости и определяет их в одном месте, что позволяет разработчикам просто использовать их, вместо того, чтобы изобретать колесо каждый раз, когда они создают новое приложение.

## **Angular**

представляет фреймворк от компании Google для создания клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. В этом плане Angular является наследником другого фреймворка AngularJS. В то же время Angular это не новая версия AngularJS, а принципиально новый фреймворк.

Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее.

Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript.

# **Часть 2. Практика**

## **2.1. Постановка задачи**

В данной курсовой работе были поставлены следующие задачи:

* разработать веб приложение (front и back);
* в качестве back cвой сервер (Springboot);
* в качестве front (Angular);
* создать Базу данных;
* Реализовать ORM подход для работы с БД используя CRUD(create, read, update, delete) методы.

## **2.2. Описание предметной области**

Ведение учета клиентов – Это приложение для хранения и управления всеми данными клиента. Admin может добавлять новых клиентов, удалять старых и искать клиентов по имени.

## **2.3. Проектирование информационной системы**

Основными сущностями для хранения информации являются:

* Customer

Также необходимо выделить атрибуты сущностей. Атрибуты — это существенные свойства объекта, интересующие пользователя. Каждая из введенных сущностей определяется набором свойств (атрибутов), которые позволяют отличать конкретные экземпляры.

Можно выделить атрибуты:

* Customer: Id,active,age,name.

На основе модели разработана структура базы данных, представленная на рисунке 2.

Graphical user interface

Description automatically generated with low confidence

Рисунок 1 — Структура базы данных

Основываясь на разработанной структуре данных, можно составить подробное описание соответствующих структур данных.

В таблице 1 представлена структура данных Product(товар), в которой хранится информация о товаре.

Таблица 1 – структура данных Customer

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Описание | Тип | Обязательное поле |
| id | Идентификатор | Целое | Да |
| Active | статус клиента | Бит | Да |
| age | возраст | Целое | Да |
| name | имя клиента | строка | Да |

## **2.4. Разработка приложения**

Взаимодействие базы данных, развёрнутой на базе Mysql, с приложением организовано средствами языка Java.

В программе реализован подход (ORM), при котором на основе существующей базы данных разрабатываются классы, сопоставляющиеся с соответствующими таблицами базы. Эти классы можно изменять по своему усмотрению, добавляя в них необходимые CRUD-методы. CRUD — четыре базовые функции, используемые при работе с хранилищами данных: create, read, update, delete.

## **Spring Boot Server**

Diagram

Description automatically generated

# **Заключение**

В ходе данной курсовой работы были закреплены навыки работы с моделью клиент-сервер, приобрел навыки работы с Java, научился применять технологии Spring Framework и Angular.

# **Результаты выполнения работы**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

# **Список литературы**

1. <https://o7planning.org/ru/11267/spring-boot-tutorial-for-beginners#a5292673>

2. <https://topjava.ru/blog/introducing-spring-boot>

3. [https:// https://metanit.com/web/angular2/1.1.php](https://habr.com/ru/post/435144/)

4. <http://crypto.pp.ua/2013/06/sozdanie-prilozhenij-klient-server-v-java/>

5.  [https://programmingtechie.com/2019/09/11/lets-build-a-simple-blog-using-spring-boot-and-angular-part-2/](%20https://programmingtechie.com/2019/09/11/lets-build-a-simple-blog-using-spring-boot-and-angular-part-2/)

6.  [https://bushansirgur.in/angular-2-and-spring-mvc-simple-crud-application/](%20https://bushansirgur.in/angular-2-and-spring-mvc-simple-crud-application/)

# основной программный код (Spring Boot Server)

## Приложение А – Dependency

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.javasampleapproach</groupId>

<artifactId>spring-boot-rest-mysql</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringRestMySQL</name>

<description>Demo project for Spring Boot - REST -MySQL</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

## Приложение Б- Клиент – модель

package com.javasampleapproach.springrest.mysql.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "customer")

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id

@Column(name = "name")

private String name;

@Column(name = "age")

private int age;

@Column(name = "active")

private boolean active;

public Customer() {

}

public Customer(String name, int age) {

this.name = name;

this.age = age;

this.active = false;

}

public long getId() {

return id;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public void setAge(int age) {

this.age = age;

}

public int getAge() {

return this.age;

}

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

@Override

public String toString() {

return "Customer [id=" + id + ", name=" + name + ", age=" + age + ", active=" + active +"]";

}

}

## Приложение В - Конфигурация для источника данных Spring и свойств JPA

spring.datasource.url=jdbc:mysql://localhost:3306/testdb?useSSL=false

spring.datasource.username=root

spring.datasource.password=fokadaama1

spring.jpa.generate-ddl=true

## Приложение Г - REST Controller

package com.javasampleapproach.springrest.mysql.controller;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.javasampleapproach.springrest.mysql.model.Customer;

import com.javasampleapproach.springrest.mysql.repo.CustomerRepository;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/api")

public class CustomerController {

@Autowired

CustomerRepository repository;

@GetMapping("/customers")

public List<Customer> getAllCustomers() {

System.out.println("Get all Customers...");

List<Customer> customers = new ArrayList<>();

repository.findAll().forEach(customers::add);

return customers;

}

@PostMapping(value = "/customers/create")

public Customer postCustomer(@RequestBody Customer customer) {

Customer \_customer = repository.save(new Customer(customer.getName(), customer.getAge()));

return \_customer;

}

@DeleteMapping("/customers/{id}")

public ResponseEntity<String> deleteCustomer(@PathVariable("id") long id) {

System.out.println("Delete Customer with ID = " + id + "...");

repository.deleteById(id);

return new ResponseEntity<>("Customer has been deleted!", HttpStatus.OK);

}

@DeleteMapping("/customers/delete")

public ResponseEntity<String> deleteAllCustomers() {

System.out.println("Delete All Customers...");

repository.deleteAll();

return new ResponseEntity<>("All customers have been deleted!", HttpStatus.OK);

}

@GetMapping(value = "customers/age/{age}")

public List<Customer> findByAge(@PathVariable int age) {

List<Customer> customers = repository.findByAge(age);

return customers;

}

@PutMapping("/customers/{id}")

public ResponseEntity<Customer> updateCustomer(@PathVariable("id") long id, @RequestBody Customer customer) {

System.out.println("Update Customer with ID = " + id + "...");

Optional<Customer> customerData = repository.findById(id);

if (customerData.isPresent()) {

Customer \_customer = customerData.get();

\_customer.setName(customer.getName());

\_customer.setAge(customer.getAge());

\_customer.setActive(customer.isActive());

return new ResponseEntity<>(repository.save(\_customer), HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

}