Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет**

**при Правительстве Российской Федерации»**

**(Финансовый университет)**

Департамент анализа данных и машинного обучения

Пояснительная записка к курсовой работе по дисциплине

«Современные технологии программирования»

на тему:

**«Разработка клиент-серверного приложения для магазина цифровой техники с использованием Spring и Java»**

Выполнил:

Студент группы ПИ19-4

Теплов А. А.

Научный руководитель:

доцент, канд. техн. наук

Петросов Д. А.

**Москва**

**2021**

Содержание

[ВВЕДЕНИЕ 3](#_Toc70719185)

[1. Постановка задачи 4](#_Toc70719186)

[2. Описание предметной области 6](#_Toc70719187)

[3. Актуальность автоматизации 6](#_Toc70719188)

[4. Описание программы 7](#_Toc70719189)

[4.1. Алгоритмические решения 7](#_Toc70719190)

[4.2 Описание интерфейса программы 12](#_Toc70719191)

[4.3. Состав приложения 20](#_Toc70719192)

[5. Назначение и состав классов программы 23](#_Toc70719193)

[ЗАКЛЮЧЕНИЕ 27](#_Toc70719194)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ 28](#_Toc70719195)

[ПРИЛОЖЕНИЕ. 1. ИСХОДНЫЙ КОД ПРОГРАММЫ 29](#_Toc70719196)

[Сервер 29](#_Toc70719197)

[Клиент 68](#_Toc70719198)

# **ВВЕДЕНИЕ**

Выполнение курсовой работы способствует приобретению навыков в области создания полноценного пользовательского приложения на основе объектно-ориентированного языка программирования Java в среде разработки IntelliJ IDEA. Java – один из самых известных среди разработчиков язык, который используется для создания приложений с пользовательским интерфейсом. Написание курсовой работы поможет усовершенствовать навыки в написании кода на данном языке. Кроме того, можно углубить знания в анализе предметной области и в визуальном проектировании постановленной задачи. Цифровые технологии стремительно развиваются, и все больше сфер нуждаются в обработке большого объема данных. Сейчас трудно представить стабильно функционирующую компанию или любое другое предприятие, которое бы не использовало автоматизированные информационные системы.

Эти системы стали основой для функционирования информационной деятельности во всех отраслях, которые начинаются от школ и поликлиник и заканчиваются крупными фабриками и атомными электростанциями. Даже средства домохозяйств стали контролироваться с использованием систем автоматизации. Вот почему данная тема актуальна.

Целью описываемой работы является разработка клиент-серверного приложения для магазина цифровой техники с использованием Spring и Java. Для успешного выполнения работы, были расставлены следующие приоритеты:

* анализ в области Информационно-Справочных систем магазинов

цифровой техники;

* постановка задачи;
* разработка базы данных для Информационно-Справочной системы;
* выбор инструментальных средств программной реализации;
* программная реализация Информационно-Справочной системы.

1. **Постановка задачи**

В соответствии с выбранной темой, требуется разработать кросс-платформенное клиент-серверное приложение для магазина цифровой техники с использованием Spring и Java, с помощью которого можно будет:

* создавать и обрабатывать заказы;
* вести учет товаров на складе;
* вносить корректировки в заказ;
* обновлять статус заказа;
* создавать записи о Клиентах;
* вносить изменения в записи о Клиентах;
* отменять заказа;
* сохранять данные в БД.

Для комфортной работы с информационной системой должен быть реализован эргономичный и интуитивно понятный графический пользовательский интерфейс.

Уточнение к требованиям:

1. Требуется разработать кросс-платформенное приложение (Клиент).
2. Необходимо приложение, в котором будут обрабатываться данные и сохраняться в БД (Сервер).
3. У приложения-клиента должен быть графический интерфейс.
4. Клиент должен обращаться к Серверу при помощи API запросов.
5. Пользовательский интерфейс приложения-клиента должен быть реализован на языке программирования Java (в данной работе интерфейс реализуется с помощью JavaFX).
6. В приложении-клиенте должно быть реализовано несколько форм пользовательского интерфейса.
7. Позволено самостоятельно определять интерфейс приложения и его функциональность, но для получения хорошей оценки вне зависимости от предметной области, должны выполняться следующие операции:

* таблице должны отображаться данные предметной области;
* таблицы для БД должны быть предварительно заполнены данными;
* в БД должно быть реализовано добавление, редактирование и удаление объекта;
* реализовать фильтрацию записей;
* реализовать сортировку записей;
* производить обновление изменений источника данных в БД;
* создать в меню раздел «Об авторе».

1. Программа ни в коем случае не должна завершаться аварийно. Необходимо производить обработку таких ошибок, как: сообщение о некорректном вводе данных, недопустимые или противоречивые значения данных, если запрашиваемые в функциональном запросе пользователя данные отсутствуют и в других нештатных ситуациях должно отображаться соответствующее сообщение в окне.
2. В программе должны присутствовать содержательные комментарии, с помощью которых можно будет автоматически сгенерировать документацию, используя инструмент JavaDoc.
3. **Описание предметной области**

К деятельности, осуществляемой магазинами цифровой техники, относятся учёт и продажа электроники различных категорий и брендов. Такие магазины всегда предоставляют обширный каталог товар и для того, чтобы их учёт был максимально точным и простым – необходимо использовать информационные системы учёта. Работникам магазина необходимо знать о наличии товара на складе, а также подробное описание, если у покупателя появятся вопросы об интересующим его товаре. Кроме того, должен вестись учёт по заказам: способ оплаты, способ получения и дата заказа, чтобы клиент не ждал свою покупку слишком долго. Актуальность заказа, чтобы отмененные или выполненные заказы не перекрывали только созданные заказы. Должны отражать сотрудники в заказе, чтобы на основе их продаж формировать финальную заработную плату.

1. **Актуальность автоматизации**

Создание информационной системы для магазина цифровой техники дает пользователям возможность с быстротой и легкостью находить всю необходимую информацию о имеющихся товарах. Товары могут быть отфильтрованы по категориям, чтобы максимально упростить поиск ноутбука, мобильного телефона или товара другой категории. Без информационной системы ведения учёта было бы затруднительно выполнять такие задачи, так как объём информации слишком большой. Разработанная программа имеет эргономичный и интуитивно понятный графический интерфейс. С её помощью можно комфортно вести учёт товаров в магазине, а также обрабатывать поступающие от покупателей заказы. Одной из главных особенностей таких систем выступает их постоянная актуальность, так как приложения может усовершенствоваться и получать новые функции в последующих обновлениях.

1. **Описание программы**
   1. **Алгоритмические решения**

В PostgreSQL базе данных находится информация о товарах, которые продаются в магазине. Кроме того, в базе данных хранится информация и наличии товара на складе, информация о клиентах и заказах, которые создаются оператором магазина. Информация о сотрудников магазина также находится в базе данных, но в отличии данных о клиенте, данные о сотрудниках не могут редактироваться самими работниками.

О каждом товаре в информационной системе имеется обязательная для каждого товара информация, а также информация, которая может относиться только к товаром определенной категории. Так, к «общей» относится следующая информация:

* название товара;
* цена товара;
* тип товара;
* серия товара;
* особенность товара;
* вес товара;
* габариты;
* категория товара;
* бренд товара.

Для информации по категориям используется такая информация, как:

* цвет товара;
* сокет процессора;
* частота процессора или видеокарты;
* технологический процесс чипа;
* разрешение экрана или матрицы;
* объем хранилища.

Магазин занимается продажей товаров, следовательно в информационной системе должны находиться покупатели и информация о них. Для повышения комфорта при работе с клиентами о них хранится информация такого содержания:

* номер телефона;
* имя покупателя;
* фамилия покупателя;
* адрес электронной почты;
* город проживания;
* улица проживания и дом проживания;
* номер квартиры;
* накопленные бонусы по программе лояльности.

Данная информация хранится в сущности «Customer» и связывается с сущностью «Orders», в которой хранится информация о заказе. В заказе располагается информация о том, кто оформил заказ, дата и время заказа, какой способ оплаты будет использоваться при расчете за товар и каким способом товар будет получен. Таким образом может быть сформирован заказ, который будет оплачен наличными при получении, а получен будет посредством курьерской доставки. Кроме того, в заказе располагается информация о сотруднике, который его сформировывал.

Не стоит забывать и тот факт, что все заказы создаются работниками магазина, о которых также нужно хранить информацию. О работниках макази хранится такая информация:

* адрес электронной почты;
* имя сотрудника;
* фамилия сотрудника;
* номер личного телефона;
* заработная плата;
* дата приема на работу;
* должность.

Заказ и товар связываются между собой при помощи сущности «OrderedItem», в которую поступает информация по заказу от «Orders», а от «Item» информация о товаре, таким образом формируется подробная информация о заказе.

Для хранения всей вышеперечисленной информации о товаре была создана сущность «Item». Все товары существуют только в базе, и пользователь не имеет к ним прямого доступа. Чтобы пользователь мог ознакомиться с товарами (увидеть их цену, вес и остальные характеристики), используется склад, для которого был создана сущность «Inventory», которая выполняет функцию отображения всех имеющихся товаров. Кроме того, в данной сущности хранится информация о цене, по которой магазин закупал тот или иной товар.

Для класса «Item» были реализованы перечисленные ниже атрибуты:

* id (артикул товара, тип данных - целое число);
* name (название товара, тип данных - строка);
* price (цена товара, тип данных - вещественное число);
* type (тип товара, тип данных - строка);
* series (серия товара, тип данных - строка);
* feature (особенность товара, тип данных - строка);
* weight (вес товара, тип данных - строка);
* \_size (габариты товара, тип данных - строка);
* color (цвет товара, тип данных - строка);
* soket (сокет процессора, тип данных - строка);
* frequency (частота чипа, тип данных - строка);
* techProcess (техпроцесс чипа, тип данных - строка);
* resolution (разрешение экрана/матрицы, тип данных - строка);
* memorySize (объем хранилища, тип данных - строка);
* category\_id (идентификатор категории, тип данных – целое число);
* brand\_id (идентификатор бренда, тип данных – целое число).

Для получения атрибутов «category\_id» и «brand\_id», сущность «Item» при помощи связи «ManyToOne» обращается к сущностям «Category» и «Brand», в которых хранится вся информация о категориях и брендах соответственно в формате идентификатора и названия.

Для сущности «Inventory» были созданы такие атрибуты:

* id (идентификатор товара на складе, тип данных - целое число);
* purchasePrice (цена закупки, тип данных - вещественное число);
* item\_id (артикул товара, тип данных - целое число).

В «Inventory» идэт связь «ManyToOne» с «Item», за счет чего на складе и отображается артикул товара.

Сущность «Customer» имеет следующие атрибуты:

* id (идентификатор покупателя, тип данных - целое число);
* phoneNumber (номер телефона, тип данных - целое число);
* firstName (имя, тип данных - строка);
* lastName (фамилия, тип данных - строка);
* email (адрес электронной почты, тип данных - строка);
* city (город проживания, тип данных - строка);
* street (улица и дом проживания, тип данных - строка);
* apart (номер квартиры, тип данных - строка);
* bonus (бонусы в системе лояльности, тип данных - целое число).

Для хранения данных о работнике создана сущность «Employee» с атрибутами:

* id (идентификатор работника, тип данных - целое число);
* email (адрес электронной почты работника, тип данных - строка);
* firstName (имя работника, тип данных - строка);
* lastname (фамилия работника, тип данных - строка);
* phoneNumber (номер личного телефона, тип данных - целое число);
* salary (заработная плата, тип данных - вещественное число);
* hireDate (дата приема на работу, тип данных - строка);
* job\_id (идентификатор должности, тип данных - целое число).

Атрибут «job\_id» хранится в сущности «Job» и содержит в себе название должности и идентификатор. За счёт связи «ManyToOne» данные из «Job» в последствии отображаются в информации о сотруднике в «Employee»

Для сущности «Orders» были созданы атрибуты такие, как:

* id (идентификатор заказа, тип данных - целое число);
* dateTime (дата и время заказа, тип данных - локальная дата);
* status (статус выполнения заказа, тип данных - строка);
* customer\_id (идентификатор покупателя, тип данных - целое число);
* delivery\_type\_id (идентификатор типа доставки, тип данных - целое число);
* payment\_id (идентификатор способа оплаты, тип данных - целое число);
* employee\_id (идентификатор ответственного за заказ сотрудника, тип данных - целое число).

Сущность «Orders» связывается с сущностью «Customer» использую связь «ManyToOne» и получает доступ к данным о покупателе, который сделал заказ. Используя эту же связь «Orders» связывается с «DeliveryType» в которой находится информация о способах получения товар, на который оформлен заказ. Способы оплаты заказа запрашиваются тоже через связь «ManyToOne» из «Payment», в котором содержится информация о различные способы оплаты заказанного товара.

Для обработки заказов используется связь, которая собственных атрибутов не имеет и только выступает в роли связующего для «Orders» и «Inventory». Использую связь «ManyToOne» с «Orders» сущность получает «order\_id» и благодаря такой же связи с «Inventory» появляется «inventory\_id». Таким образом получается максимально оптимизированный и структурированный список по заказам, с которым максимально комфортно работать в разработанной информационной системе.

## **4.2 Описание интерфейса программы**

При запуске приложения отображается окно заказов (см. рис. 1), на котором отображается вся необходимая для сотрудника информация. Для создания этого в приложении Scene Builder был создан пустой AnchorPane на который был наложен SplitPane для разделения экрана на два горизонтальных окна. В верхней части был создан AnchorPane с TableView, в котором отображаются все заказы. В нижнию часть окна был до AnchorPane с GridPane и TableView. На GridPane показывается находится информация о покупателе, а в TableView информация о заказанных товарах.

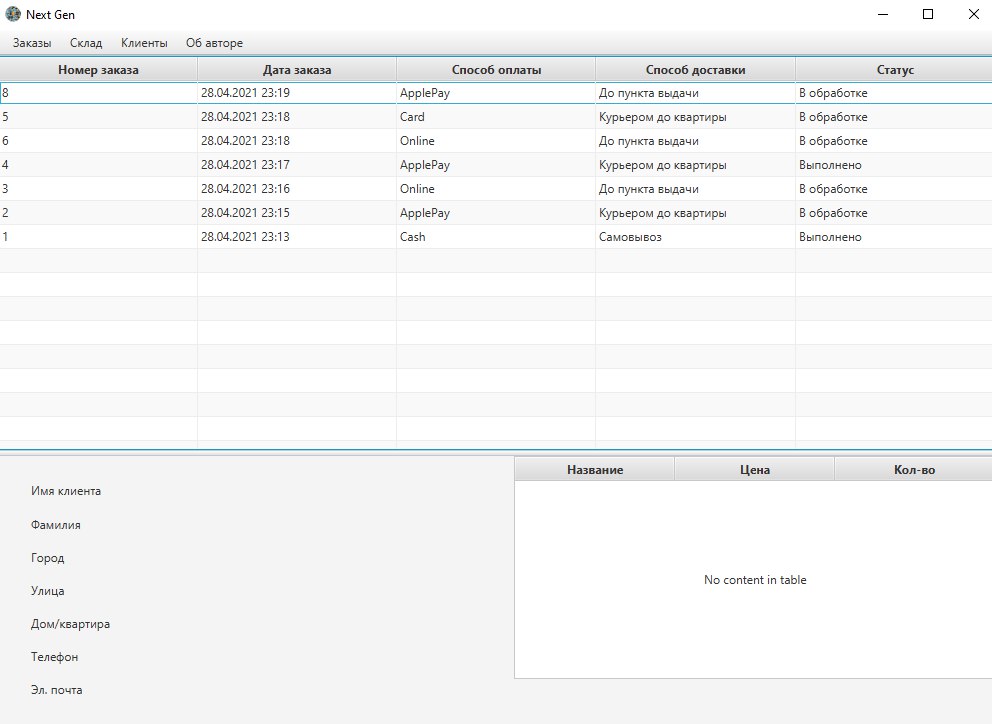


Рисунок 1 - Окно заказов

Нажатии на произвольный заказ (см. рис. 2) отобразит подробную информацию о нем. В GridPane появляется отображение данных о покупателе, в нижнем правом TableView отображается название товара его количество, а также цена. В правом нижнем углу присутствует Label, который отображает общую сумму заказа.

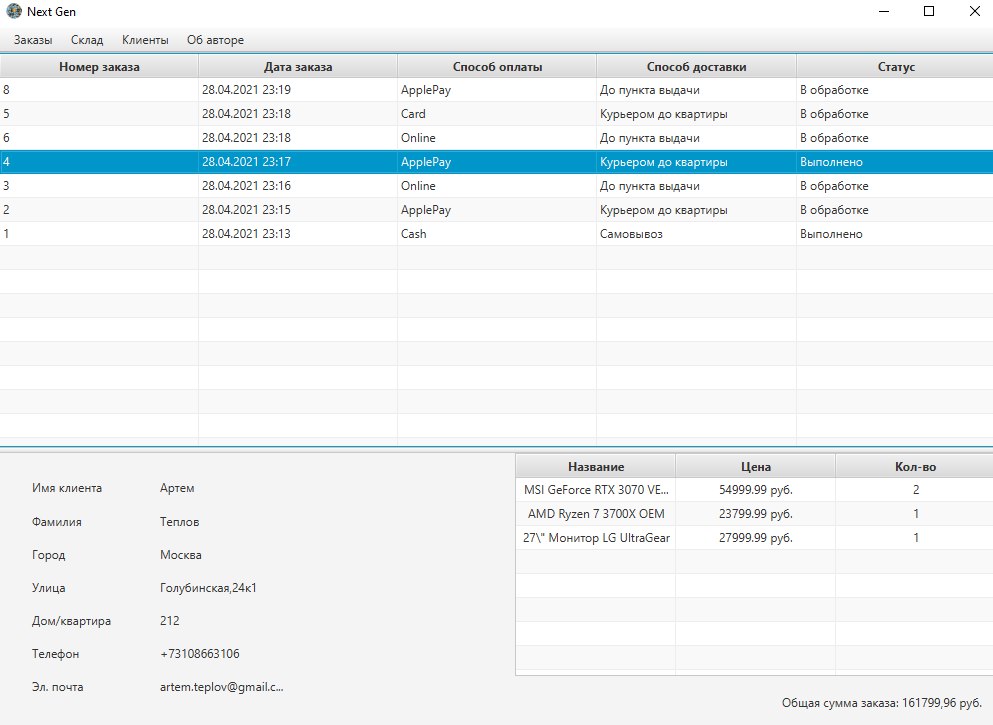


Рисунок 2 - Окно заказов с отображенной информацией

При нажатии на правую кнопку мыши около курсора появляется контекстное меню (см. рис. 3), в котором пользователю предоставляется возможность пометить заказ как выполненный, отменить его, внести корректировки в заказ, а также создать новый заказ.



Рисунок 3 - Контекстное меню

Перейдя по пункту контекстного меню «Добавить», открывается окно добавления заказа (см. рис. 4). С помощью SplitPane экран разделяется на список товаров и данные о заказе. Отображение товаров реализовано при помощи TableView, заполнение данных осуществляется в GridPane.

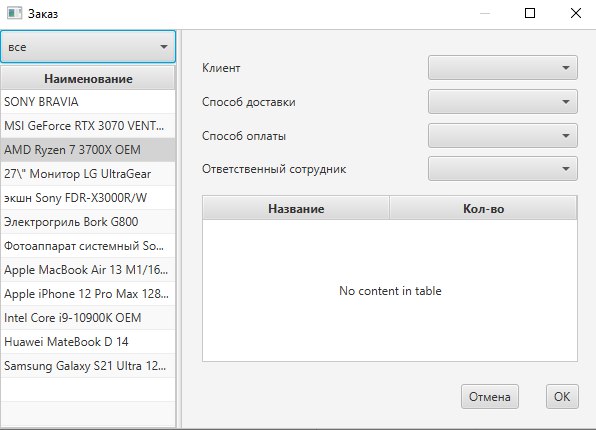


Рисунок 4 - Окно настройки заказа

Кроме того, в данном окне есть возможность отфильтровать товары по категориям (см. рис. 5), при помощи ComoBox.

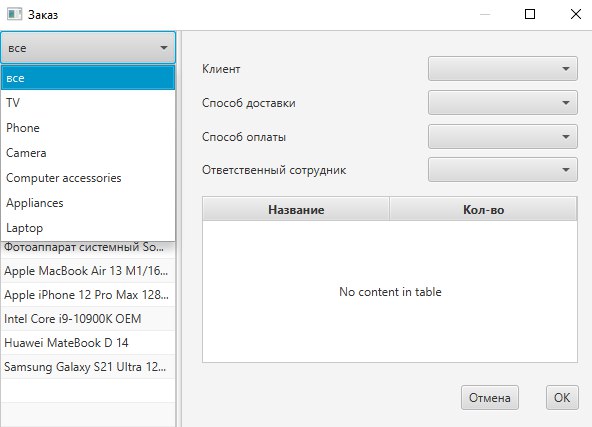


Рисунок 5 - Фильтрация товаров по категориям

Одинарное нажатие левой кнопкой по категории мыши вызывает список всех товаров, которые относятся к выбранной категории. Если попытаться создать пустой заказ, то программа выведет сообщение ошибки (см. рис. 6), которое проинформирует о том, что какое-то из окон не было заполнено.

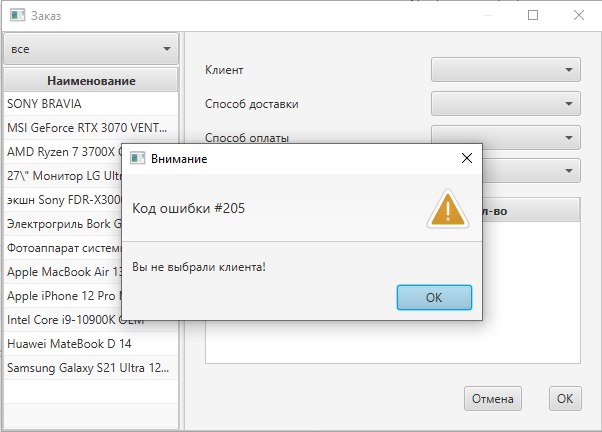


Рисунок 6 – Сообщение об ошибке при оформлении заказа

Так же в сообщении может говориться о том, что в заказ не был добавлен товар, не был выбран способ доставки товара, отсутствует способ оплаты товара или не указан сотрудник, который оформлял данный заказ.

Для наиболее комфортного заполнения информации о заказе, было использовано множество ComboBox, через которые очень удобно заполнять такие шаблонные поля (см. рис. 7), как способ доставки или способ оплаты. Такая реализация позволяет экономить время при оформлении заказа и избежать всевозможные опечатки, которые могли бы произойти при ручном вводе с клавиатуры. Все ComboBox расположены внутри GridPane, в котором они соотносятся со своими названиями.

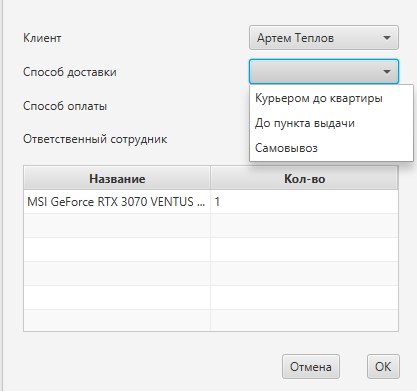


Рисунок 7 - Заполнение данных при оформлении заказа

Если во время оформление заказа был добавлен не тот товар, то при помощи контекстного меню его можно убрать (см. рис. 8).

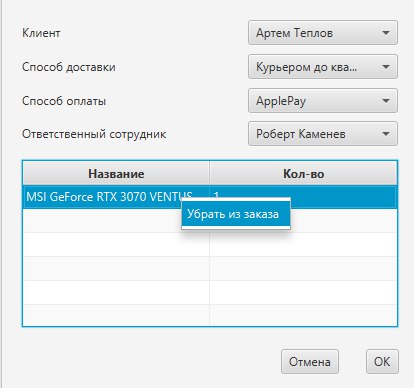


Рисунок 8 - Удаление товара из заказа

Для получения подробной информации о каждом товаре было реализовано окно склада (см. рис. 9). Оно состоит из двух, разделенных SplitPane, горизонтальны секций. С TableView для перечесления всех имеющихся в магазине товаров и GridPane для отображения информации о выбранном товаре.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 9 - Окно склада с информацией о товар

Для того, чтобы можно было быстро находить интересующий товар, в окне склада был реализован поиск (см. рис. 10).



Рисунок 10 - Поиск товаров на складе

Пользователь имеет возможность производить поиск товара как по наименованию, так и по его артикулу.

Третья вкладка в пользовательском интерфейсе отвечает за информацию о покупателях. Окно разделено на TableView и GridPane при помощи SplitPane. При нажатии на имя или фамилию в TableView на GridPane появляется подробная информация о покупателе (см. рис. 11).

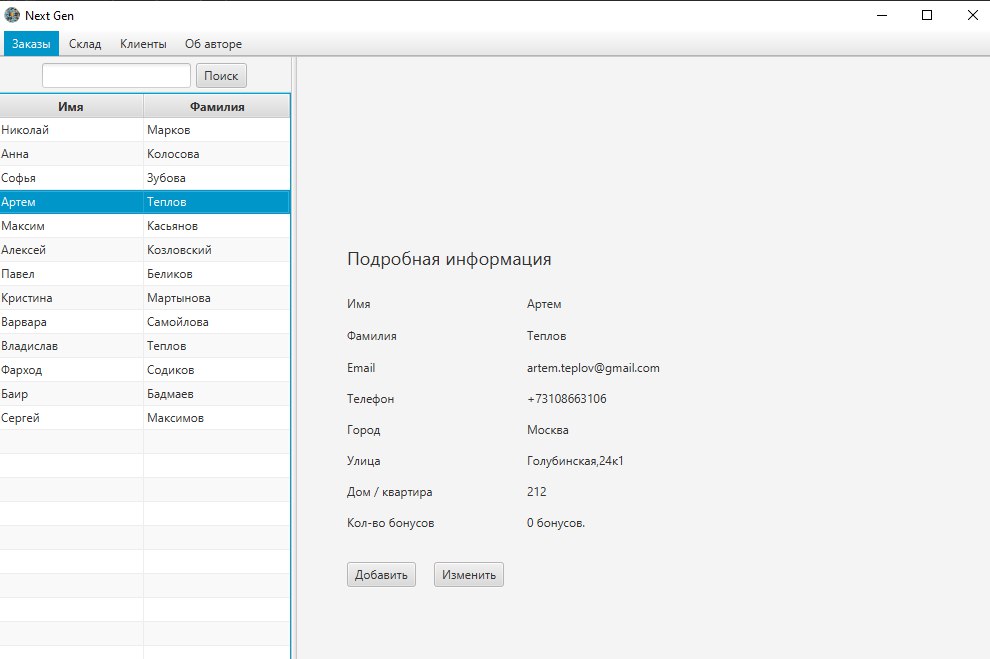


Рисунок 11 - Информация о клиентах магазина

При объемной базе клиентов может возникнуть проблему с поиском нужного человека. С целью решения данной проблемы был добавлен поиск по номеру мобильного телефона (см. рис. 12), так как в данном случае он является уникальным для каждого покупателя.

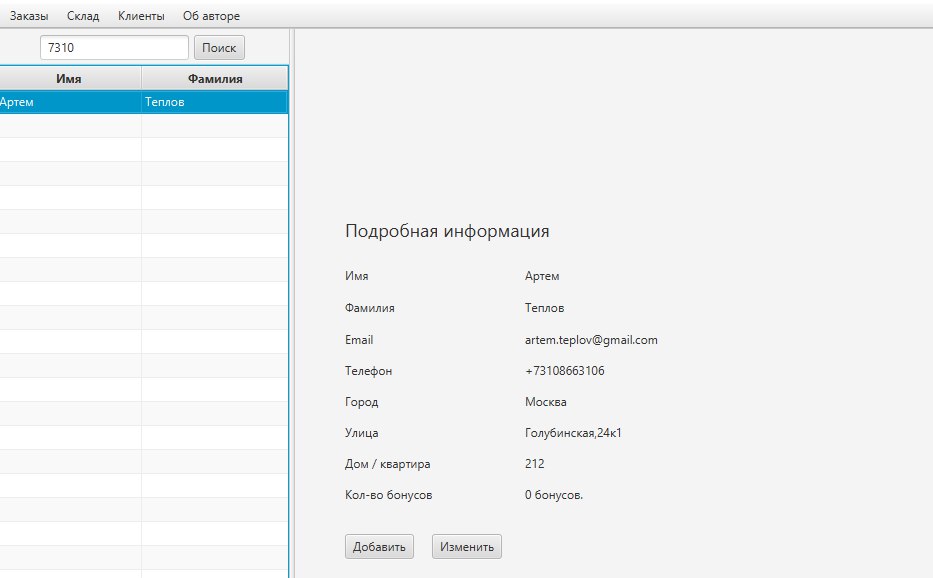


Рисунок 12 - Поиск клиента по номеру телефона

Если у покупателя произошли изменения в адресе проживания или любой другой личной информации, то программой предусмотрена возможность внесения корректировки в данные пользователя (см. рис. 13).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 – Внесение изменений в личную информацию покупателя

С помощью аналогичной формы можно вносить в БД новых клиентов.

Финальная вкладка «Об авторе» (см. рис. 13) содержит в себе один Label в котором просто написаны имя, фамилия и наименование учебной группы автора программы.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 - Окно "Об авторе"

Это было всё описание интерфейса реализованной программы.

* 1. **Состав приложения**

Приложение состоит из трех основных компонентов:

* сервер;
* клиент;
* база данных.

Для реализации сервера были использованы следующие компоненты:

* Lombok – удобный набор аннотация, который помогает сокращать шаблонный код;
* Spring Web – позволяет создавать веб-приложения, в том числе RESTful, с использованием Spring MVC (Model-View-Controller). Использует Apache Tomcat в качестве встроенного контейнера по умолчанию;
* Spring Data JPA – сохраняет данные в хранлищах SQL при помощи Java Persistence API с использованием Spring Data;
* Spring Boot DevTools – обеспечивает быстрый перезапуск приложения, загрузку и конфигурацию для улучшения опыта разработки.
* Apache maven – выступает в роли основного сборщика для реализованного проекта;
* Spring-boot-maven-plugin – плагин для сборки решения.

При создании клиента были использованы такие компоненты и внешние библиотеки, как:

* JavaFX (javafx-controls, javafx-fxml, javafx-graphics) – библиотека для отображения графического пользовательского интерфейса и её основные компоненты;
* Google GSON – модуль для работой с запаковкой данных в программе, а также для их распаковки;
* OkHttp – удобный способ для реализации обмена данными между сервером и клиентом, так как имеет высокую скорость закрузки. Поддерживаются как синхронные блокирующие вызовы, так и асинхронные.

В качестве СУБД используется PstgreSQL в которой были созданы следующие коллекции и поля:

* brand – c полями id и title;
* category – с полями id и title;
* customer – с полями id, apart, bonus, city, email, first\_name, last\_name, phone\_number и street;
* delivery\_type – с полями id и title;
* employee – с полями id, email, first\_name, hare\_date, last\_name, phone\_number, salary и job\_id;
* inventory – с полями id, purchase\_price и item\_id;
* item – с полями id, \_size, color, feature, frequency, memory\_size, name, price, resolution, series, soket, texh\_process, type, weight, brand\_id и category\_id;
* job – с полями id и title;
* orders – с полями id, amount, date\_time, status, customer\_id, delivery\_type\_id, employee\_id и payment\_id;
* orders\_inventories – с полями orders\_id и inventories\_id;
* payment – с полями id и title;

Дополнительные пояснения по каждой коллекции:

* brand – коллекция брендов продаваемых товаров;
* category – коллекция категорий продаваемых товаров ;
* customer – коллекция с подробной информацией о каждом клиенте, которого занесли в БД;
* delivery\_type – коллекция осуществляемых способов доставок заказанных товаров;
* employee – коллекция с подробной информацией о всех сотрудника в магазине;
* inventory – коллекция с подробной информацией продаваемых в настоящий момент товарах;
* item – коллекция с подробной информацией о товарах, их характеристиках и цене;
* job – коллекция возможных должностей для сотрудников магазина;
* orders – коллекция с информацией о сформированном заказе;
* orders\_inventories – коллекция для объединяющая информацию;
* payment – коллекция возможных способов оплаты заказа.

Для более комфортного понимания работы реляционной базы данных ниже представлена подробная схема (см. рис. 15).



Рисунок 15 - Реляционная модель БД

На выше представленной схеме подробно показаны все связи, а также первичные и внешние ключи для этих связей.

1. **Назначение и состав классов программы**

Сервер

Для корректной работы приложения на сервере задействуется 34 класса, которые разбиваются по следующим группам:

* контроллеры (8 штук) - классы для обработки входящих запросов от клиентов;
* сущности (10 штук) - данные, передаваемые сервером к клиенту;
* репозитории (8 штук) – наборы стандартных методов JPA для работы с БД;
* сервисы (8 штук) – классы, которые нужны для обработки полученных на контроллере данных. Выступают в качестве посредника между контроллерами и репозиториями.

В сервер входят следующие классы:

* CategoryController - контроллер для http запросов, связанных с категориями товаров;
* CustomerController - контроллер для http запросов, связанных с данными о покупателе;
* DeliveryTypeController - контроллер для http запросов, связанных с реализуемыми способами получения товара;
* EmployeeController - контроллер для http запросов, связанных с данными о сотрудниках;
* InventoryController - контроллер для http запросов, связанных с товарами находящимися на складе;
* ItemController - контроллер для http запросов, связанных с продаваемыми товарами;
* OrderController - контроллер для http запросов, связанных с оформлением заказа;
* PaymentController - контроллер для http запросов, связанных с возможными способами оплаты покупки;
* Brand - сущность для предоставления разновидности брендов товаров в СУБД;
* Category - сущность для предоставления разновидности категорий товаров в СУБД;
* Customer - сущность для предоставления пользователя товаров в СУБД;
* DeliveryType - сущность для предоставления разновидности способов получения заказа в СУБД;
* Employee - сущность для предоставления сотрудников в СУБД;
* Inventory - сущность для предоставления товаров на складе в СУБД;
* Item - сущность для товаров в СУБД;
* Job - сущность для предоставления разновидности должностей в СУБД;
* Orders - сущность для предоставления заказов в СУБД;
* Payment - сущность для предоставления разновидности способов оплаты заказов в СУБД;
* CategoryRepository - интерфейс JpaRepository для работы с коллекцией категориями товаров;
* CustomerRepository - интерфейс JpaRepository для работы с коллекцией покупателей;
* DeliveryTypeRepository - интерфейс JpaRepository для работы с коллекцией типов доставки;
* EmployeeRepository - интерфейс JpaRepository для работы с коллекцией сотрудников;
* InventoryRepository - интерфейс JpaRepository для работы с коллекцией товаров на складе;
* ItemRepository - интерфейс JpaRepository для работы с коллекцией товаров;
* OrderRepository - интерфейс JpaRepository для работы с коллекцией заказов;
* PaymentRepository – интерфейс JpaRepository для работы с коллекцией типами оплаты заказов;
* CategoryService - сервис для работы с категориями товаров;
* CustomerService - сервис для работы с покупателями;
* DeliveryTypeService - сервис для работы с типами доставок;
* EmployeeService - сервис для работы с сотрудниками;
* InventoryService - сервис для работы с товарами на складе;
* ItemService - сервис для работы с товарами;
* OrderService - сервис для работы с заказами;
* PaymentService - сервис для работы с типами оплаты заказов.

Клиент

Для клиента было создано 17 классов, которые были разделены на такие группы, как:

* класс работы с API (1 штука) - содержит логику для общения с сервером;
* классы контроллеры (7 штук) - обрабатывают действия пользователя на формах пользовательского интерфейса;
* классы сущности (10 штук) - данные, передаваемые клиентом серверу.

В клиенте реализованы следующие классы:

* API - класс с реализацией методов Gson и Okhttp3;
* CustomerDialogController - контроллер-обработчик окна настройки клиентов;
* CustomersController - контроллер-обработчик окна информации о клиентах;
* OrderDialogController - контроллер-обработчик окна настройки заказа;
* OrdersController - контроллер-обработчик окна информации о заказе;
* RootController - контроллер-обработчик вкладок заказы/склад/клиенты/об авторе;
* StorageController - контроллер-обработчик окна товаров, находящихся на складе;
* Brand - модель-сущность бренда;
* Category - модель-сущность категории товара;
* Customer - модель-сущность покупателя;
* DeliveryType - модель-сущность способа доставки;
* Employee - модель-сущность сотрудника;
* Inventory - модель-сущность склада;
* Item - модель-сущность товара;
* Job - модель-сущность должности;
* Order - модель-сущность заказа;
* Payment - модель-сущность способа оплаты.

# **ЗАКЛЮЧЕНИЕ**

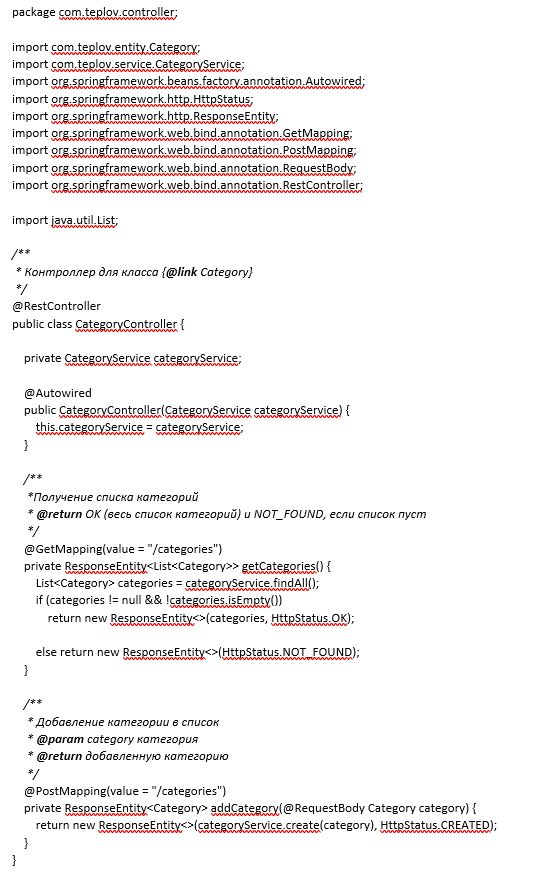
В выполненной работе все поставленные задачи и цели были успешно реализованы. Приложение работает стабильно на разных платформах без аварийных отключений. Было создано клиент-серверное приложение для магазина цифровой техники с использованием Spring и Java. С его помощью пользователи могут эффективно и с максимальным комфортом ввести учёт товаров и заказов магазине. Приложение имеет интуитивно понятный пользовательский интерфейс, который основан на всем известных приложениях, благодаря чему у конечного пользователя не возникнет вопросов при работе с ним, а, главное, не нужно проводить для пользователя обучение перед началом работы. Данная информационная система имеет возможность дальнейшего развития с добавлением изменений в пользовательский интерфейс и с введением новых функциональных особенностей. В процессе выполнения данной работы, были приобретены навыки в области разработки сложных приложений и повысился уровень написания кода. Кроме того, был получен большой опыт в понимании работы сферы торговли и зависимости этой сферы от развитых и стабильных информационно-справочных систем.

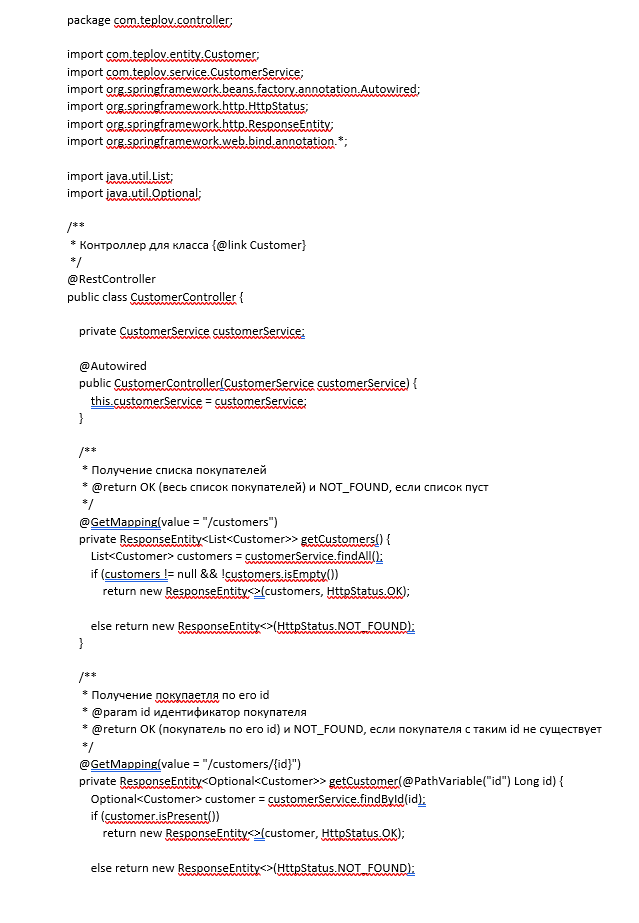
**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ**

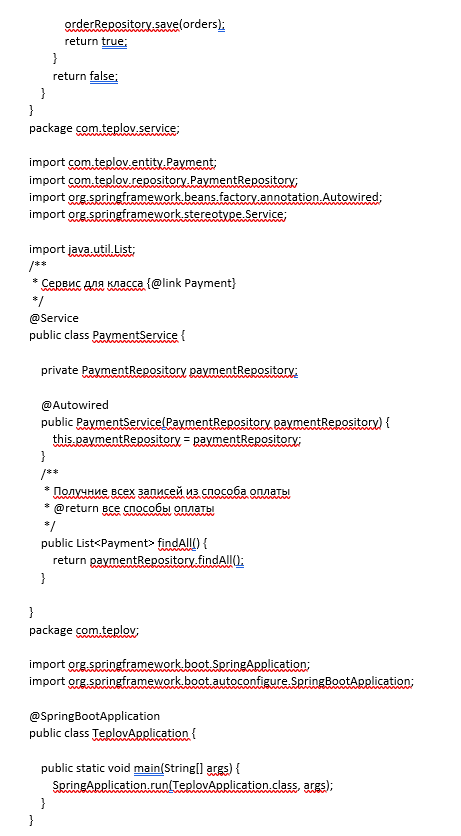
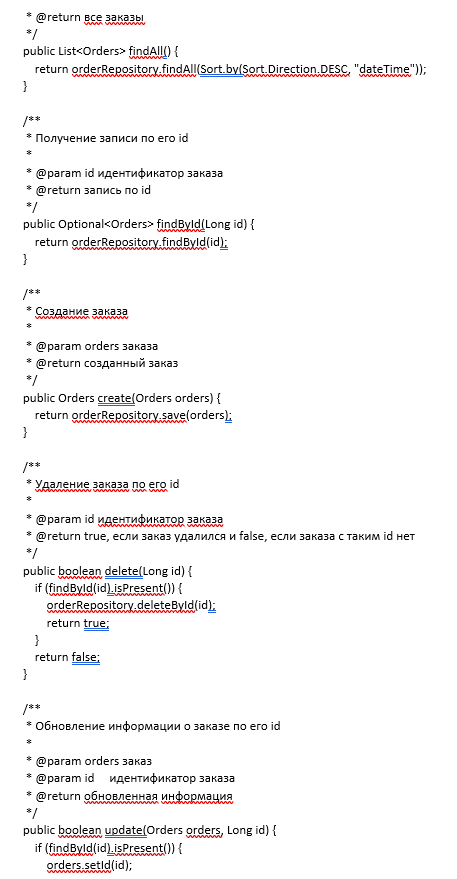
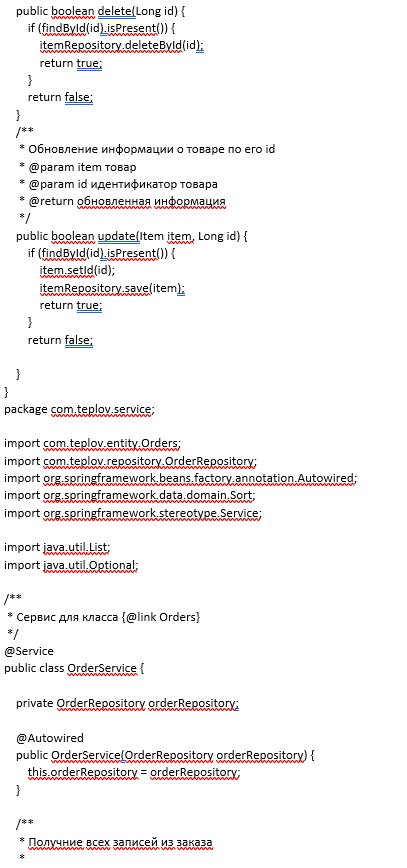
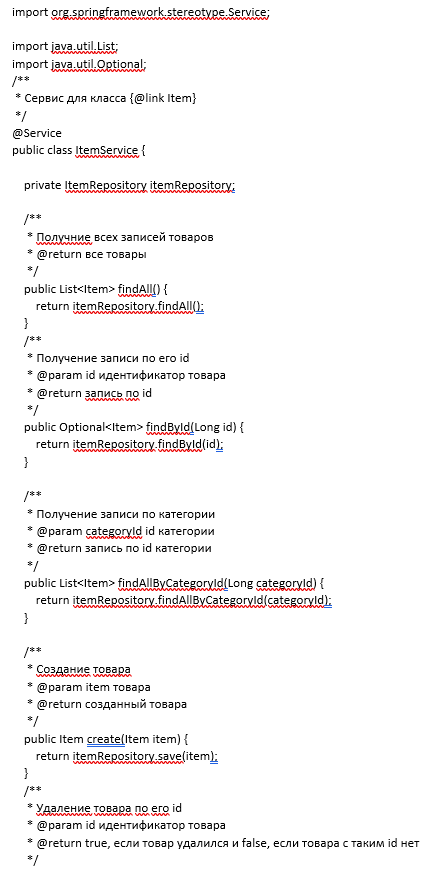
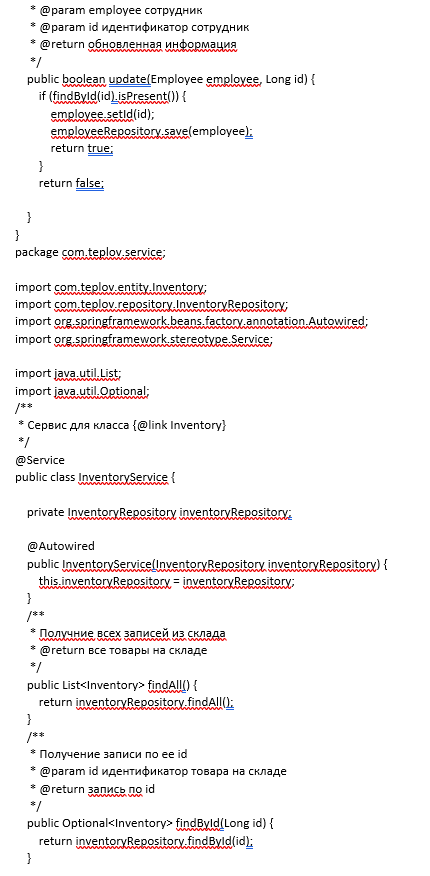
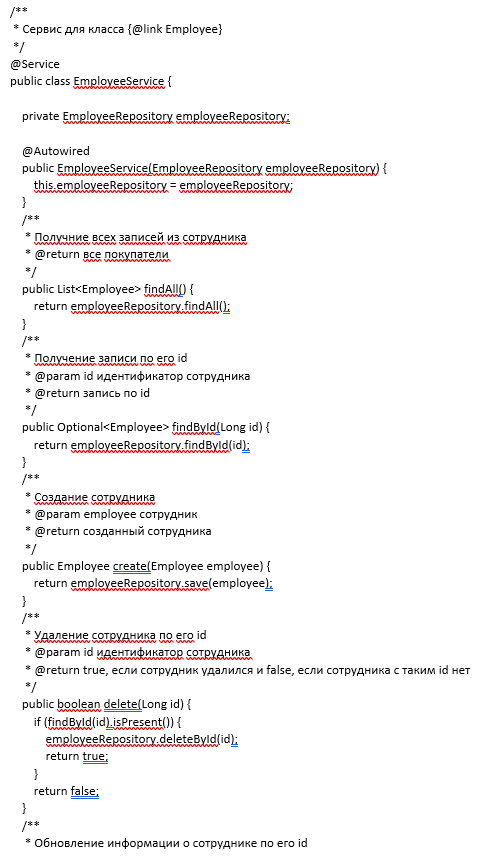
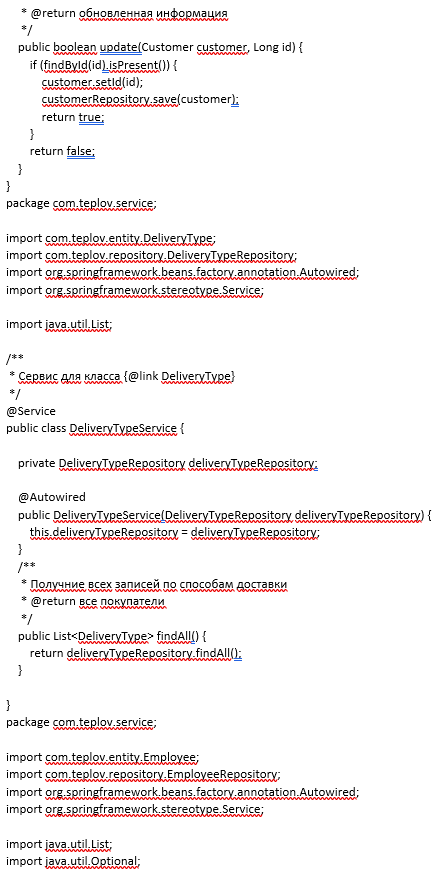
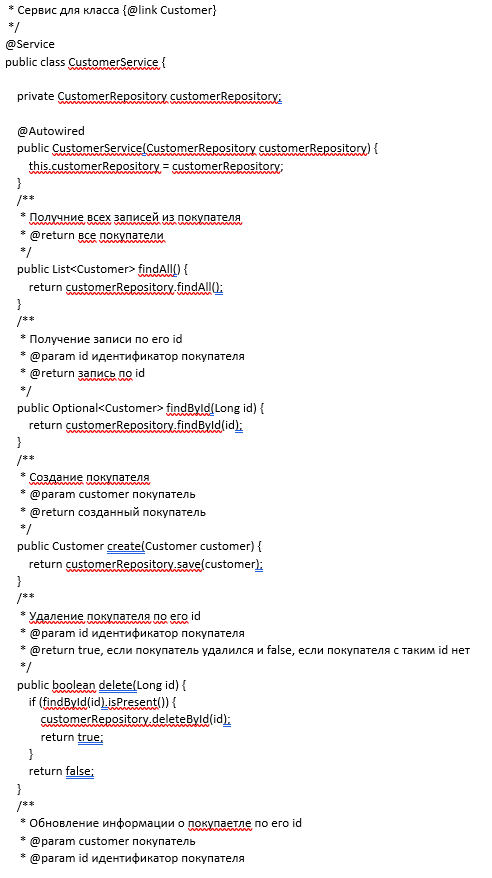
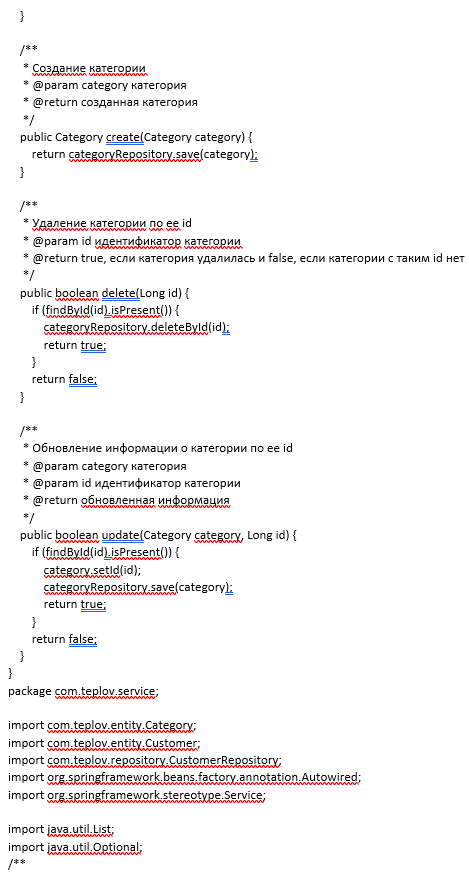
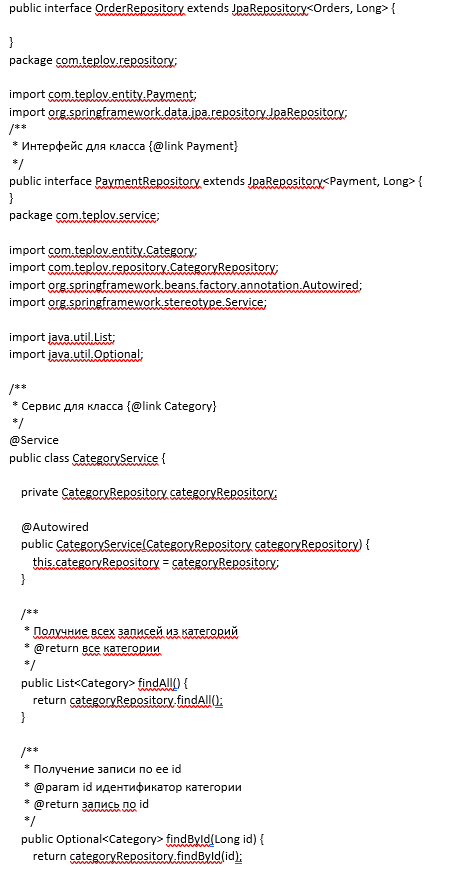
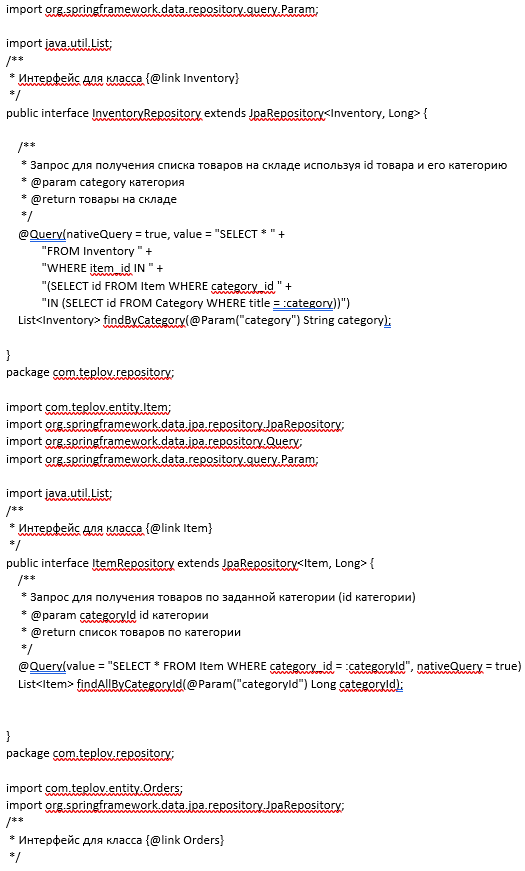
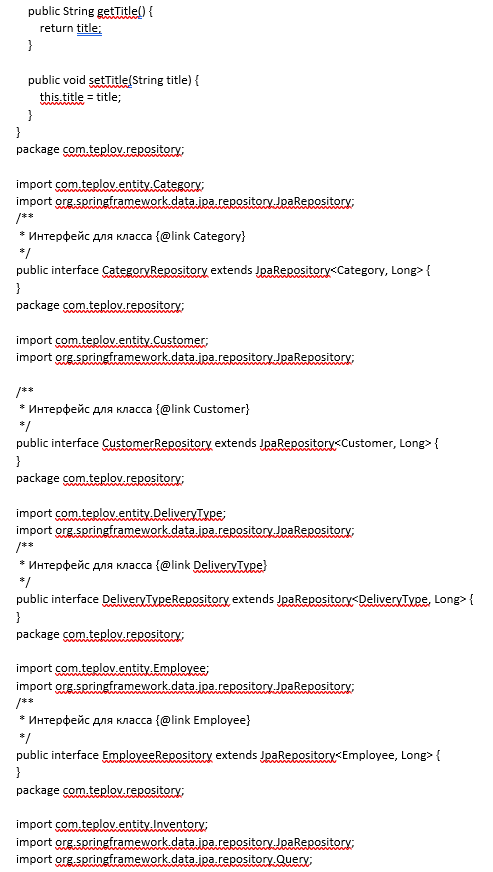
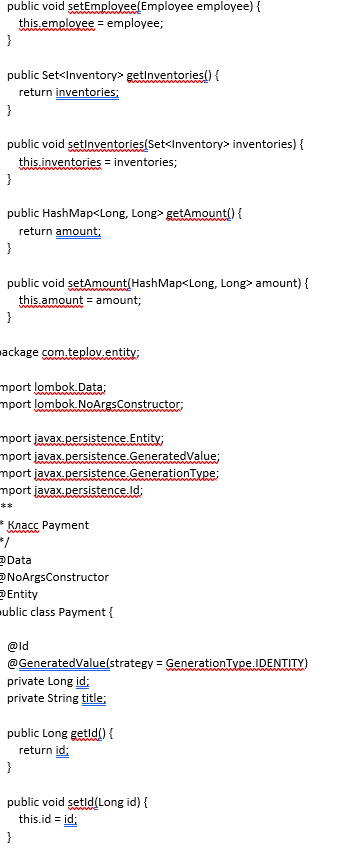
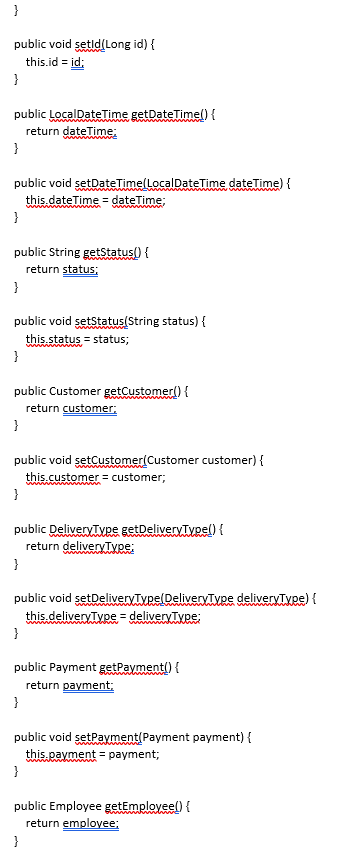
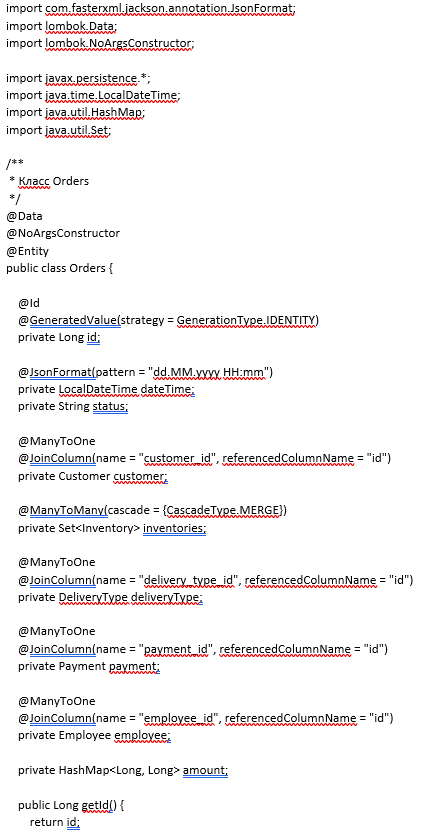
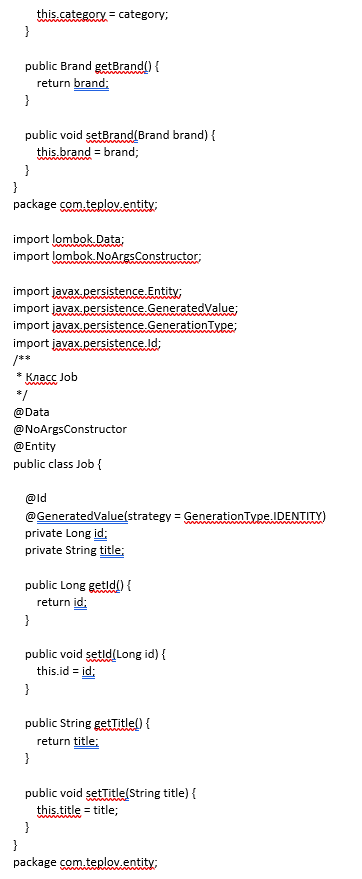
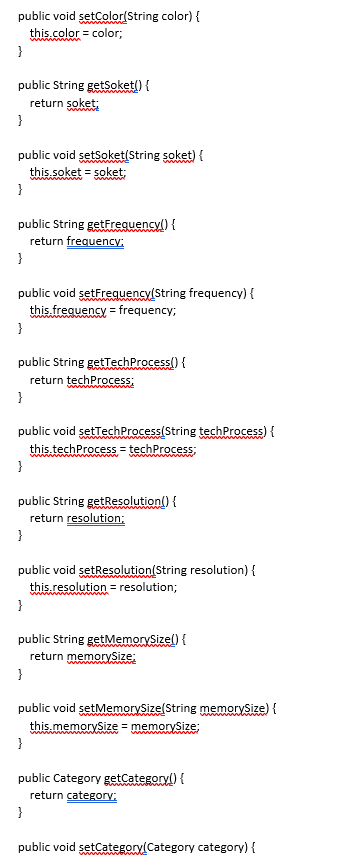
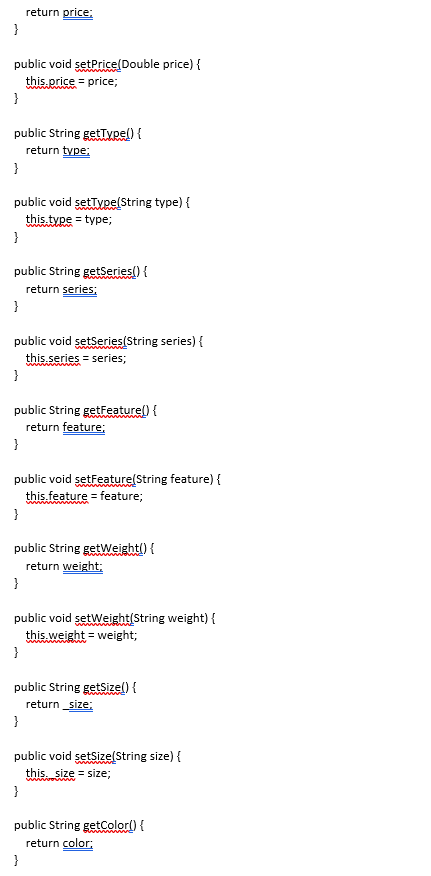
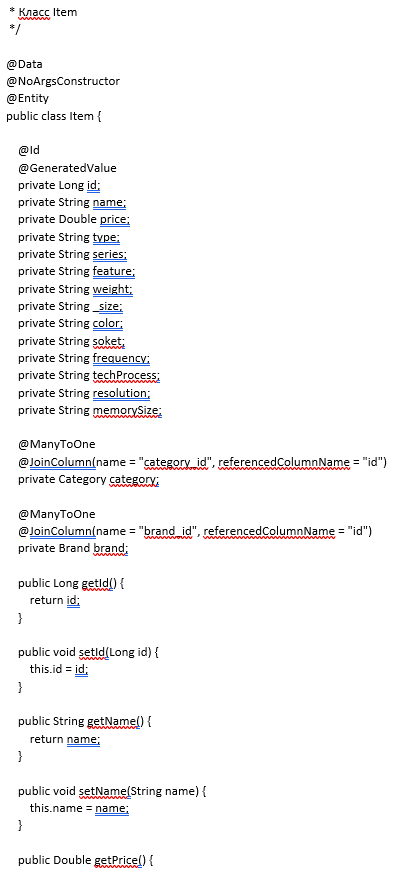
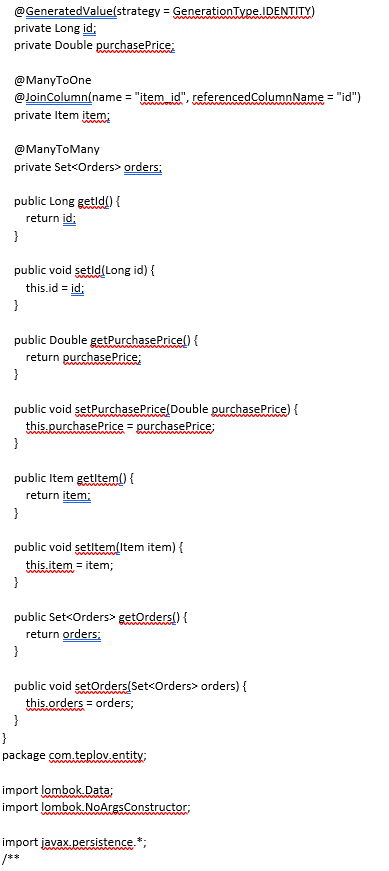
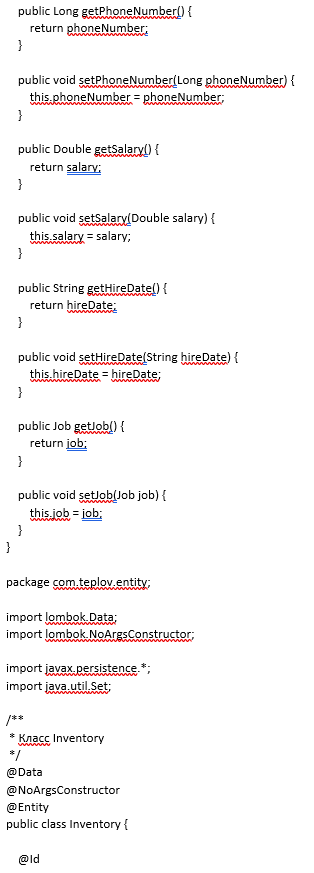
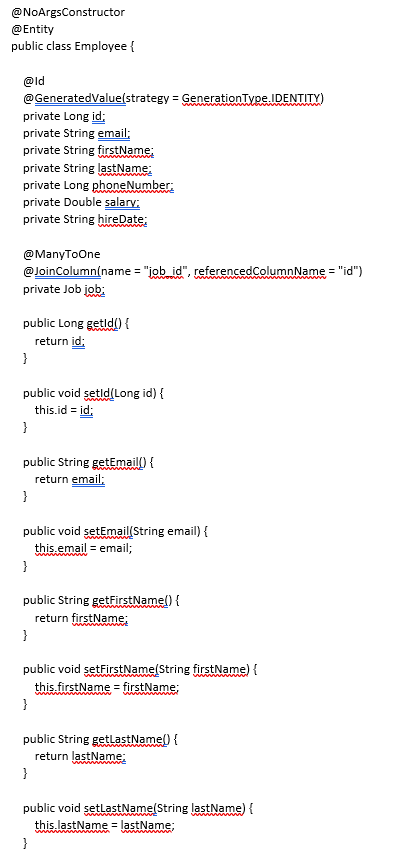
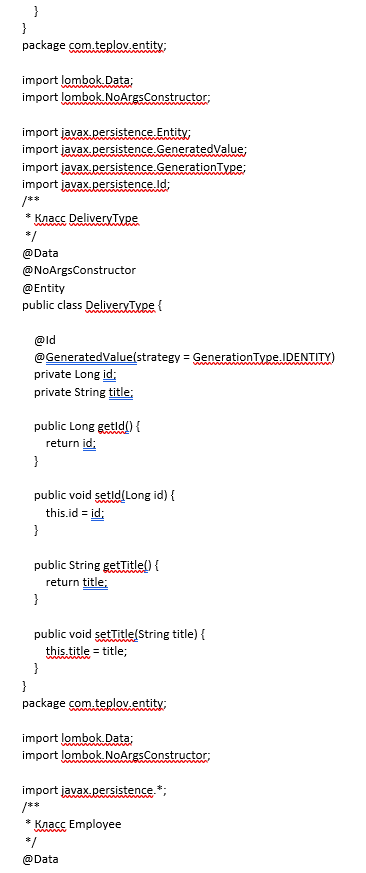
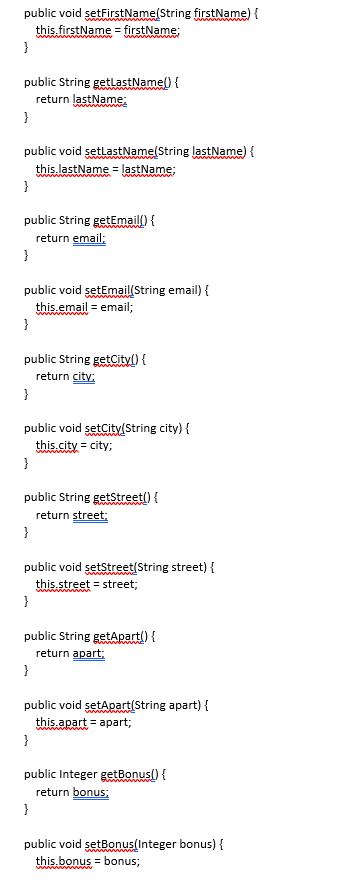
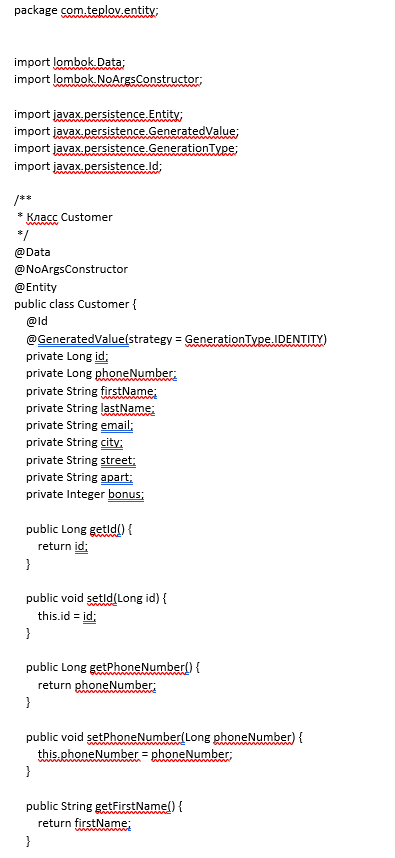
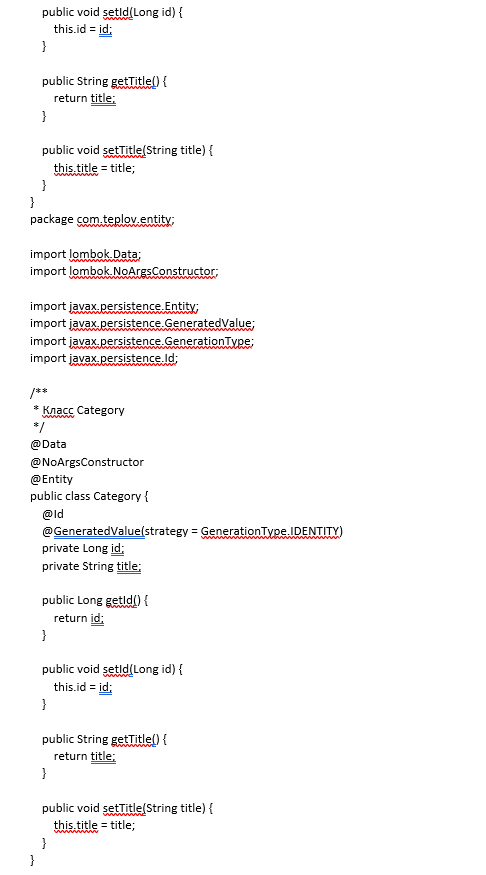
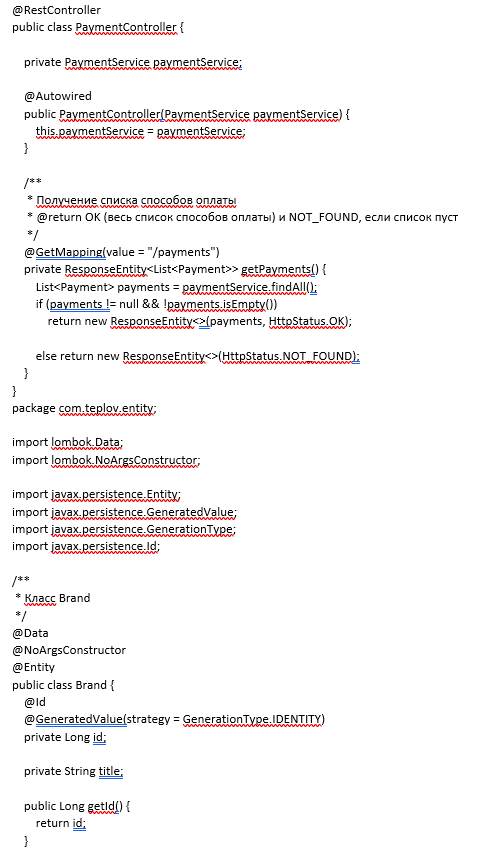
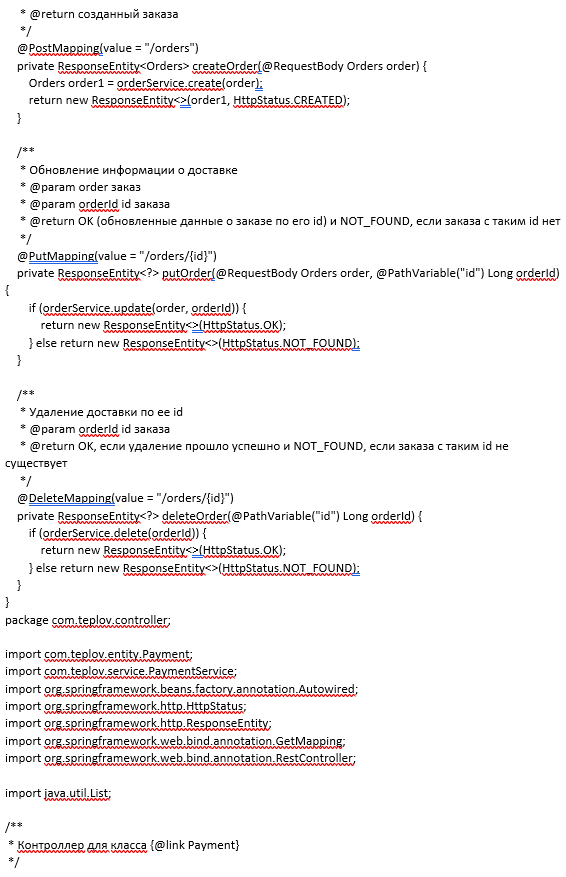
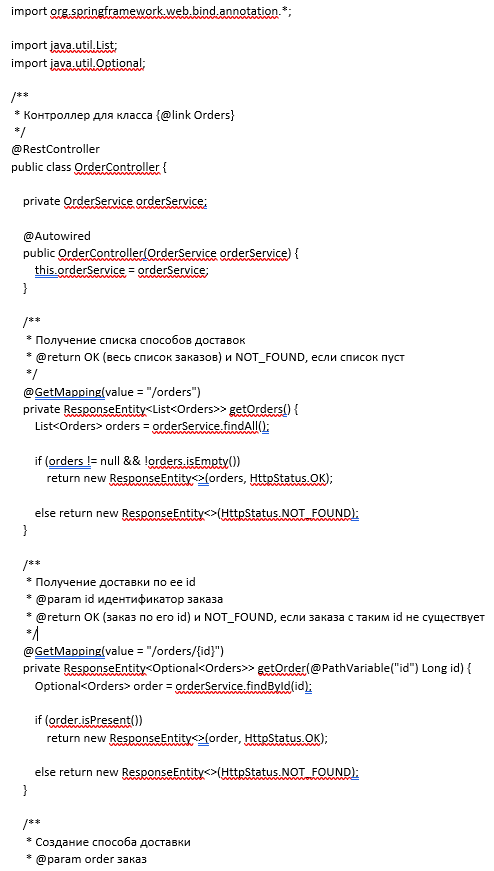
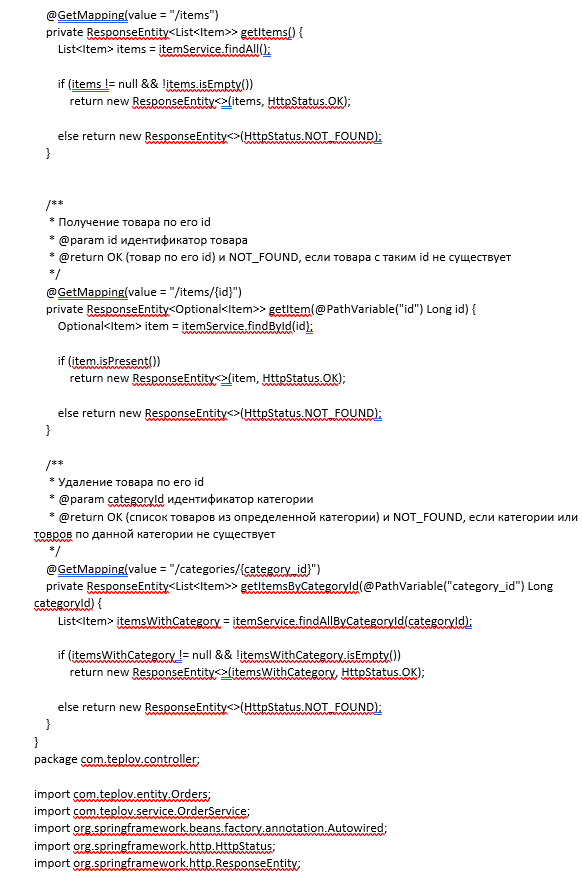
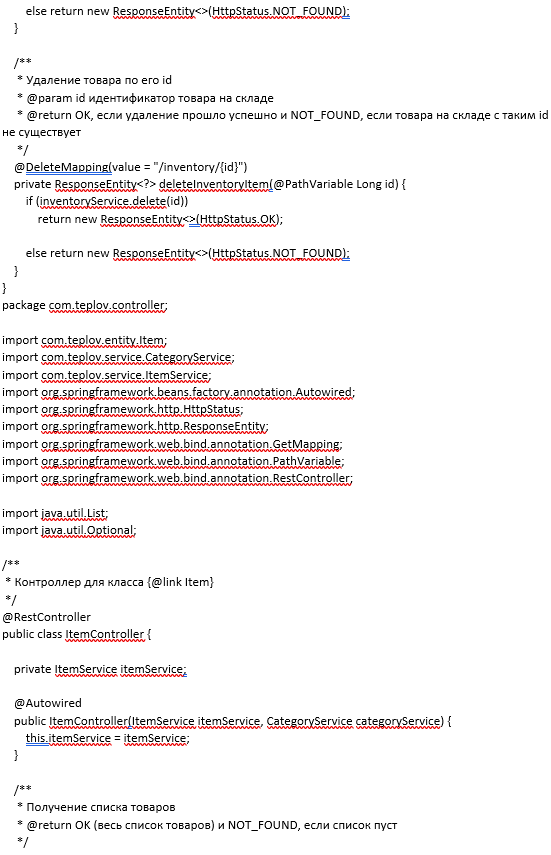
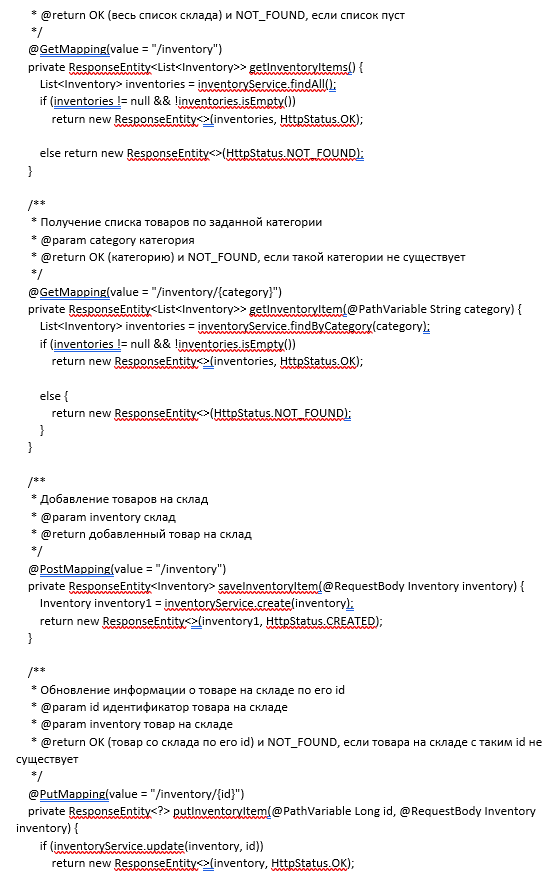
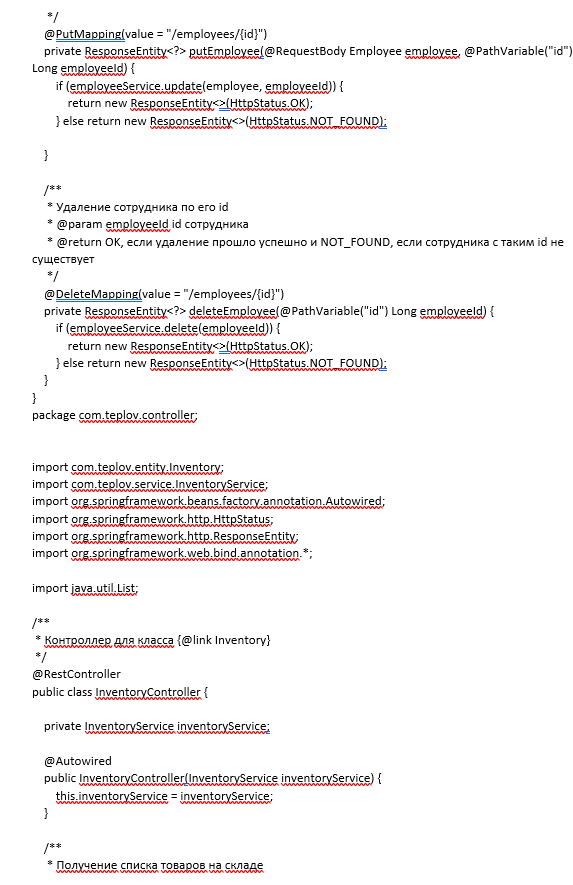
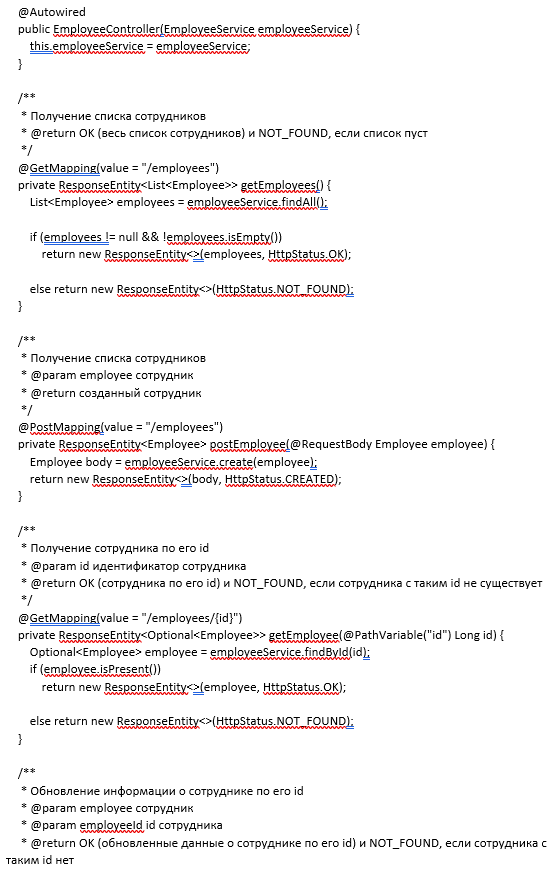
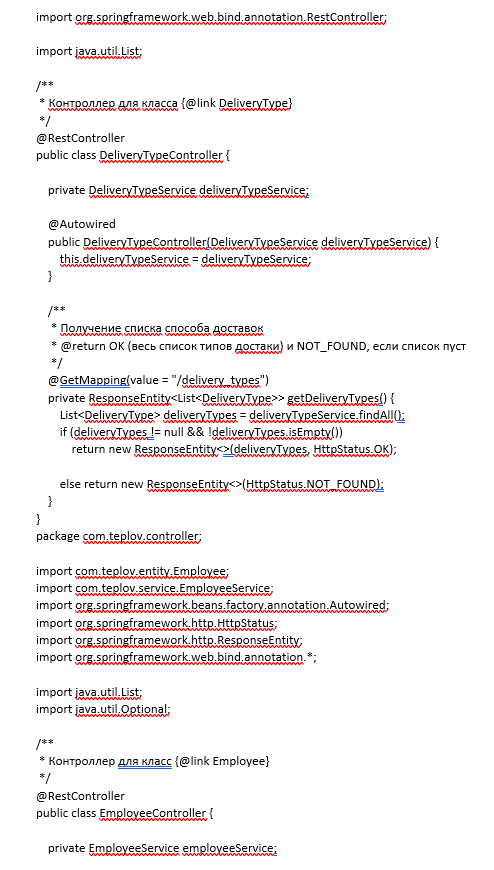
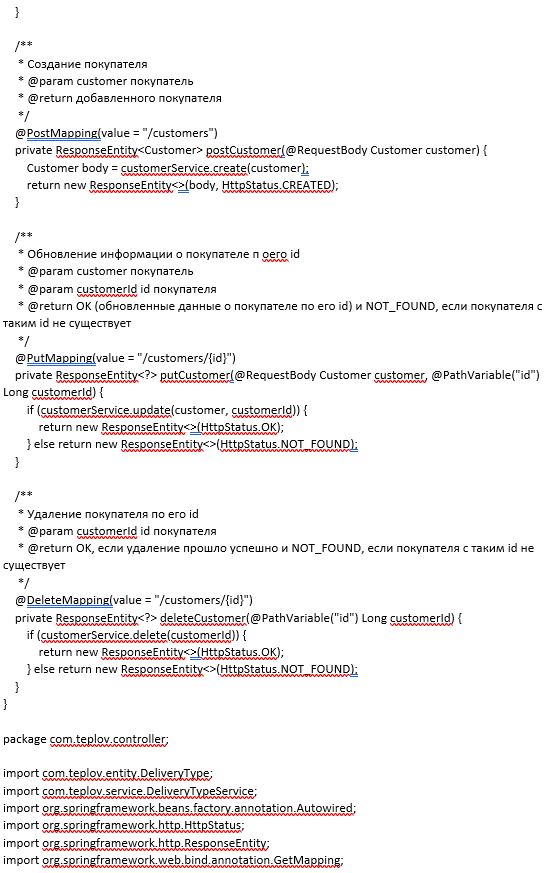
1. Козмина Ю., Харроп Р. Spring 5 для профессионалов. - Киев: Диалектика-Вильямс, 2019. - 1120 с.
2. Коузен К. Современный Java. Рецепты программирования . - М.: ДМК Пресс, 2018. - 274 с.
3. Мартин Роберт К. Чистый код. Создание анализ и рефакторинг. - СПб: Питер, 2019. - 464 с.
4. Прохоренок Н.А. JavaFX. - СПб: БХВ-Петербург, 2020. - 768 с.
5. Шилдт Г. Java. Полное руководство.-Киев: Диалектика, 2018.-1488 с.
6. Связанные сущности в Hibernate [Электронный ресурс] // java-online. 12.03.2021. URL: <http://java-online.ru/hibernate-entities.xhtml>
7. Java json parser – пример работы парсера [Электронный ресурс] // hr-vector. 20.03.2021. URL: <https://hr-vector.com/java/json-parser>

# ПРИЛОЖЕНИЕ. 1. ИСХОДНЫЙ КОД ПРОГРАММЫ

## Сервер







## Клиент

