Содержание

[Введение 3](#_Toc116823813)

[1 Сбор, анализ и формирование требований к программному продукту 4](#_Toc116823814)

[1.1 Сбор требований 4](#_Toc116823815)

[1.2 Анализ и формирование требований 5](#_Toc116823816)

[1.2.1 Общие требования 5](#_Toc116823817)

[1.2.2 Требования к функциональным характеристикам 6](#_Toc116823818)

[1.2.3 Требования к надежности 6](#_Toc116823819)

[1.2.4 Требования к квалификации и численности персонала 7](#_Toc116823820)

[2 Проектирование и разработка архитектуры программного продукта 8](#_Toc116823821)

[1.2 Построение диаграммы связей 8](#_Toc116823822)

[2.2 Разработка сценария использования 9](#_Toc116823823)

[2.4 Архитектура программного продукта 11](#_Toc116823824)

[2.4 Выбор СУБД и разработка базы данных 13](#_Toc116823825)

[2.5 Прототипирование и дизайн программного продукта 15](#_Toc116823826)

[3 Разработка программного продукта 18](#_Toc116823827)

[3.1 Инструментальные и программные средства разработки 18](#_Toc116823828)

[3.2 Календарный план разработки 21](#_Toc116823829)

[4 Тестирование программного продукта 22](#_Toc116823830)

[4.1 Выбор метода обеспечения качества 22](#_Toc116823831)

[4.2 Тестирование программного продукта 22](#_Toc116823832)

[5 Разработка документации на программный продукт 26](#_Toc116823833)

[Заключение 27](#_Toc116823834)

[Список литературы 28](#_Toc116823835)

[Приложение А 29](#_Toc97032711)

# Введение

Целью курсовой работы является разработка информационной системы с использованием принципов ООП на языке высокого уровня Kotlin в среде программирования IntelliJ IDEA - компьютерный салон.

Заказчику требуется информационная система для автоматизации работы компьютерного салона по продаже комплектующих для компьютера.

С данным предложением он обратился к нам за помощью в разработке информационной системы.

После анализа всех требований и пожеланий заказчика мы взялись за разработку информационной системы для его салона.

1 Сбор, анализ и формирование требований к программному продукту

## Сбор требований

На этапе сбора требований необходимо составить вопросы для заказчика и сформировать их в таблицу-опросник.

Таблица 1 – Интервью с заказчиком

|  |  |
| --- | --- |
| Вопросы | Ответы |
| Какие виды деятельности магазина видеопродукции необходимо автоматизировать? | Операции по оформлению поставок и списания видеопродукции, предварительный заказ, продажу компьютерных комплектующих |
| В каком формате должен быть вывод отчётов о видах деятельности салона? | В формате Excel-таблиц |
| Какую входную информацию необходимо использовать для осуществления деятельности компьютерного салона? | Необходимы таблицы с данными о реализуемой магазином товаре, о продавцах магазина и поставщиках комплектующих |
| Кто будет пользоваться программой? | Продавец магазина и системный администратор (менеджер) |
| Кто будет иметь возможность редактировать пароли пользователей? | Только администратор (менеджер) |
| Какие функции выполняет продавец? | Он осуществляет продажу товара, его резервирование, а также имеет возможность добавление товара в базу данных(бд) салона |
| Какие функции выполняет администратор (менеджер)? | Администратор оформляет на склад новую продукцию, полученную в результате поставок, имеет возможность списывать видеопродукцию со склада, а также ведет информацию по продавцам и поставщикам магазина |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Вопросы | Ответы |
| Нужны ли пользователям программы фильтры для поиска и выбора комплектующих? | Должна быть предусмотрена фильтрация информации: 1) о комплектующих (по производителю, по типу комплектующих, по поставщику, наличию на складе, резервированию); 2) поставщиков (по названию, типу комплектующих, количеству возможной поставки); |
| Будет ли доступ с компьютера в сеть? | Да, но ограниченный. Для связи с поставщиками/складом. Так же для подборки возможных комплектующих, которых нет в магазине или на складе. |

## 1.2 Анализ и формирование требований

### 1.2.1 Общие требования

Информационная система «Компьютерный сало» предназначено для автоматизации деятельности магазина, который осуществляет продажу комплектующих для компьютера.

Программа должна предоставлять информационную систему для занесения и обновления в базе данных информации о реализации магазином продукции, о его продавцах, поставщиках.

В состав технических средств, поддерживающих работу программы, должен входить IВМ-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор AMD Ryzen 3 1200 – 3.1GHz, не менее;
* оперативную память объемом 2Гигабайта, не менее;
* HDD 40 Гигабайт, не менее;
* операционную систему Windows 10 и выше.

### 1.2.2 Требования к функциональным характеристикам

Информационная система «Компьютерный салон» должно обеспечивать возможность выполнения перечисленных ниже функций:

* предоставление продавцу возможности оформить продажу товара – каждая операция оформляется квитанцией;
* предоставление продавцу возможности быстрого поиска нужного комплектующего (по названию, по категории);
* предоставление менеджеру магазина полного доступа к хранящейся в базе данных информации с возможностью ее обновления и редактирования;
* предоставление менеджеру возможности занесения в базу данных (на склад) новой продукции;
* предоставление менеджеру возможности списывать видеопродукцию со склада (списываются комплектующие, у которых обнаружился брак);
* предоставление менеджеру возможности организации быстрого поиска комплектующих (по поставщику, типу комплектующего, по наличию на складе/магазине), поставщиков (по названию и количеству поставляемых комплектующих), продавцов (по фамилии или логину).

В программе необходимо предусмотреть возможность корректировки настроек системы; резервное сохранение данных; возможность изменения паролей входа в систему.

### 1.2.3 Требования к надежности

#### 1.2.3.1 Требования к обеспечению надежного функционирования программы

Разрабатываемое программное обеспечение должно иметь:

* парольную защиту при запуске программы;
* ограничение несанкционированного доступа к данным;
* возможность резервного копирования информационной базы;
* разграничение пользовательских прав;
* исключение несанкционированного копирования (тиражирования) программы.

Предусмотреть контроль вводимой информации и блокировку некорректных действий пользователя при работе с системой.

### 1.2.4 Требования к квалификации и численности персонала

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц — системный администратор (менеджер) и продавец.

Системный администратор должен иметь высшее профильное образование. В перечень задач, выполняемых системным администратором, должны входить:

* задача поддержания работоспособности технических средств;
* задачи установки (инсталляции) и поддержания работоспособности системных программных средств — операционной системы;
* задача установки (инсталляции) программы;
* задача создания резервных копий базы данных.

2 Проектирование и разработка архитектуры программного продукта

## 1.2 Построение диаграммы связей

Диаграмма связей программного продукта представляет собой графическую схему взаимодействия объектов (модулей, страниц и т. д.) проектируемого ПП. Схема взаимодействия программных модулей показывает структуру программы, её составные части (модули), отражает связь между ними и показывает, какое место в этой связи занимает база данных. Схема отражает не только подчиненность модулей, но и порядок их вызова или функционирования программы.

Графическое изображение схемы взаимодействия модулей информационной системы «Компьютерный салон» представлено на рисунке 1.

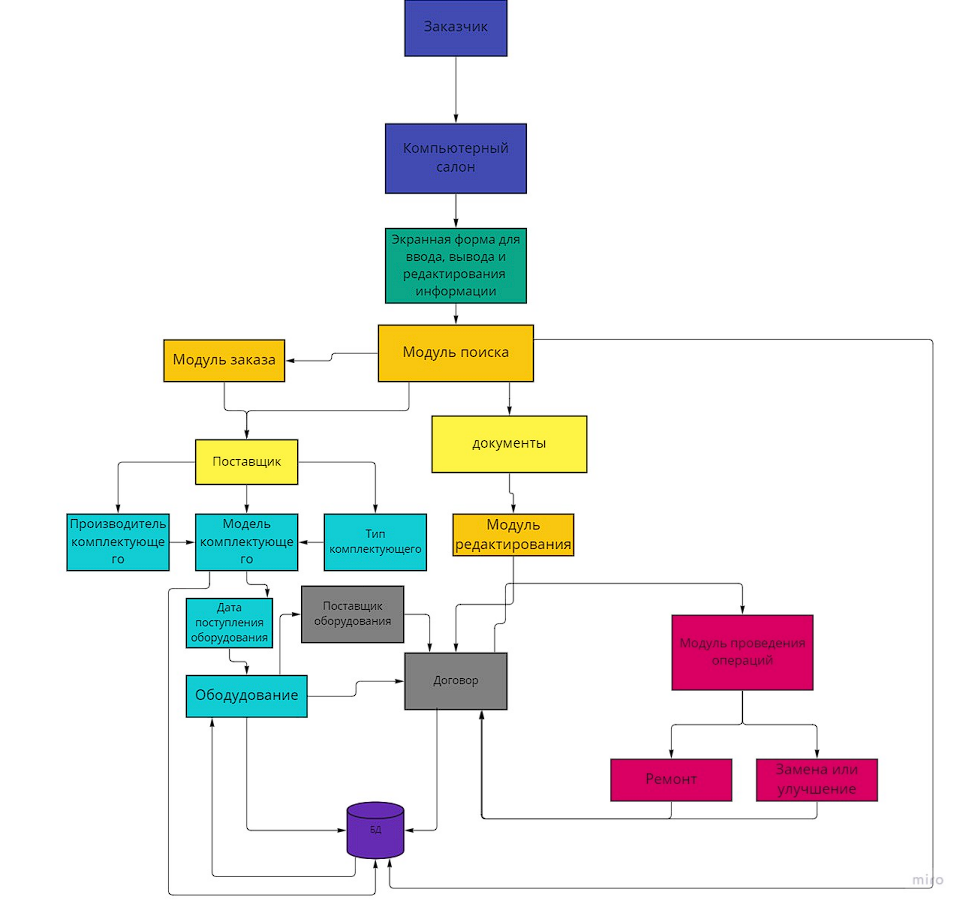


Рисунок 1 ‑ Схема взаимодействия модулей проектируемого ПП

## 2.2 Разработка сценария использования

Сценарий использования, проектируемого ПП можно разработать с помощью диаграмм вариантов использования (Use-case) и деятельности UML. Диаграмма Use-case разработана с целью определения функционала информационной системы «Компьютерный салон» для его пользователей, и представлен на рисунке 2.



Рисунок 2 ‑ Диаграмма вариантов использования, проектируемого ПП

Диаграмма деятельности (активности) UML позволяет более детально визуализировать конкретный вариант использования. Эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Диаграмма деятельности для варианта использования (функции) представлена на рисунке 3.

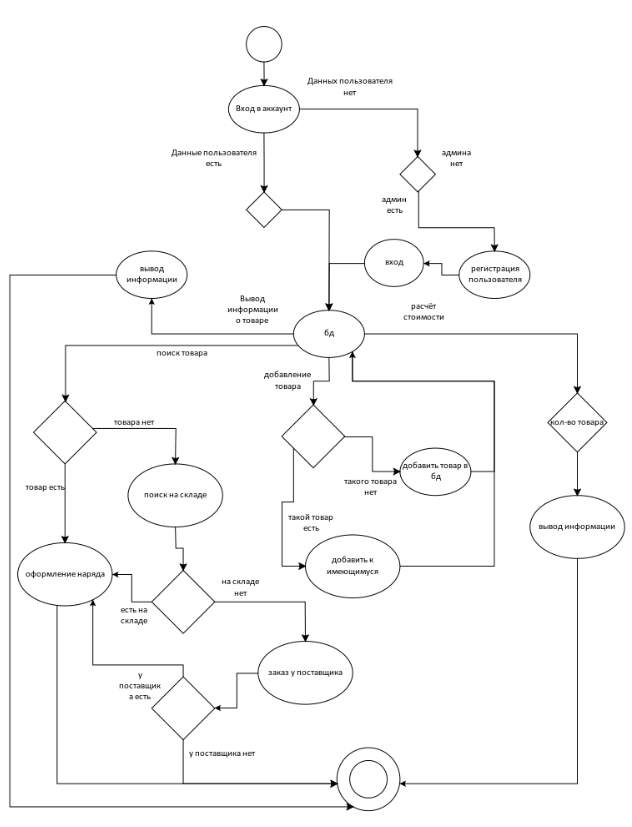


Рисунок 3 ‑ Диаграмма деятельности для функций редактирования справочников БД

## 2.4 Архитектура программного продукта

При проектировании информационной системы «Компьютерный салон» необходимо продумать его работу на нескольких компьютерах. Архитектура представлена на рисунке 4.

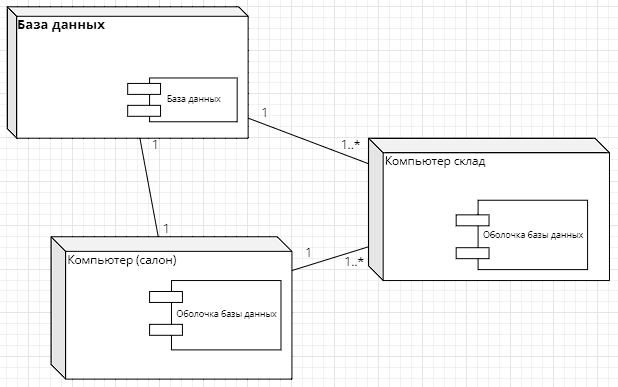


Рисунок 4

Почти всё современное программирование построено на принципах ООП и для проектирования внутренней структуры объектно-ориентированного приложения используют диаграмму классов (class diagram), предназначенную для представления программы в виде классов и связей между ними. Диаграмма классов информационной системы «Компьютерный салон» представлена на рисунке 5.

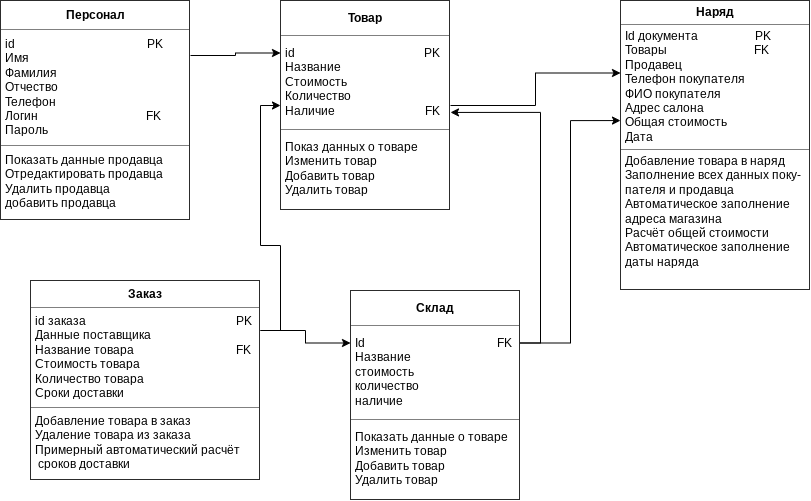


Рисунок 5 – Диаграмма классов проектируемого ПП

## 2.4 Выбор СУБД и разработка базы данных

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Существует обширный список типов данных, которые поддерживает PostgreSQL. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), PostgreSQL может похвастаться поддержкой uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов. PostgreSQL обеспечивает хранение разных типов сетевых адресов

Характеристика СУБД приводятся в таблице 2.

Таблица 2 – Характеристики СУБД

|  |  |
| --- | --- |
| СУБД | Краткая характеристика |
| PostgreSQL | Реляционная СУБД, имеет многие возможности, которые реализованы в крупных коммерческих продуктах. ОС ‑ Unix, Windows и NetWare |

Для интеграции данных в информационную систему «Компьютерный салон» была выбрана СУБД PostgreSQL. Структура спроектированной базы данных приведена в таблицах 3-7.

Таблица 3 – Персонал

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание поля |
| Код персонала | Числовой | Ключевое |
| Имя | Текстовый | Имя персонала |
| Фамилия | Текстовый | Фамилия персонала |
| Отчество | Текстовый | Отчество персонала |
| Телефон | Числовой | Телефон персонала |
| Логин | Текстовый | Внешний ключ |
| Пароль | Текстовый | Пароль персонала |

Таблица 4 – Товар

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание поля |
| Код товара | Числовой | Ключевое |
| Название | Текстовый | Название товара |
| Стоимость | Числовой | Стоимость товара |
| Количество | Числовой | Количество товара |
| Наличие | Текстовый | Внешний ключ |

Таблица 5 – Заказ

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание поля |
| Код заказа | Числовой | Ключевое |
| Данные поставщика | Текстовый | Информация о поставщике |
| Название товара | Текстовый | Внешний ключ |
| Стоимость товара | Числовой | Стоимость товара |
| Количество товара | Числовой | Количество товара |
| Сроки доставки | Дата/время | Приблизительные сроки доставки |

Таблица 6 – Наряд

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание поля |
| Код документа | Числовой | Ключевое |
| Товары | Текстовый | Внешний ключ |
| Продавец | Текстовый | ФИО продавца |
| Телефон покупателя | Числовой | Телефон покупателя |
| ФИО покупателя | Текстовый | ФИО покупателя |
| Адрес салона | Текстовый | Адрес салона |
| Общая стоимость | Числовой | Стоимость покупки |
| Дата | Дата/время | Дата покупки |

Таблица 7 – Склад

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание поля |
| Код товара | Числовой | Внешний ключ |
| Название | Текстовый | Название товара |
| Стоимость | Числовой | Стоимость товара |
| Количество | Числовой | Количество товара на складе |
| Наличие | Текстовый | Наличие товара на складе |

## 2.5 Прототипирование и дизайн программного продукта

Прототип — это набросок внешнего вида продукта. Прототип показывает, что и как действует при работе пользователя с интерфейсом.

Прототип информационной системы «Компьютерный салон», разработанный с помощью сервиса Figma, представлен на рисунках 7-10.

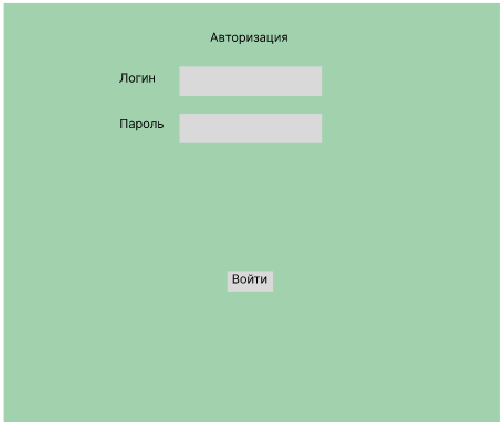


Рисунок 7 – Авторизация



Рисунок 8 – База данных магазина

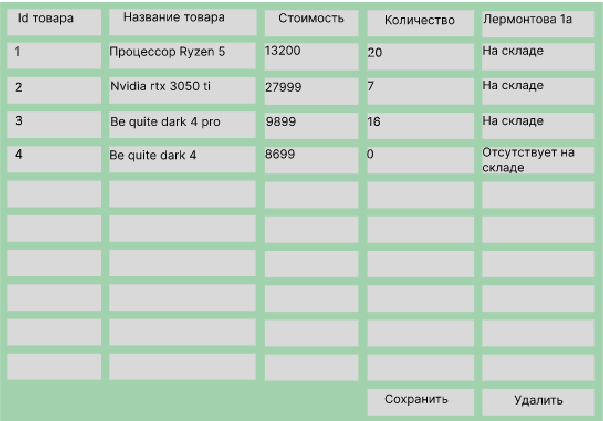


Рисунок 9 – База данных склада

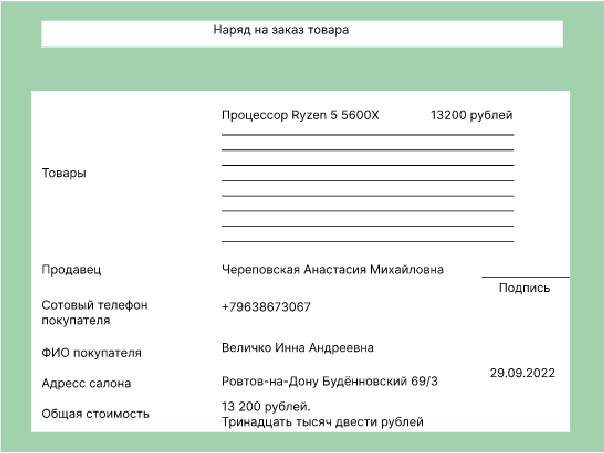


Рисунок 10 – Наряд на заказ товара

# 3 Разработка программного продукта

## 3.1 Инструментальные и программные средства разработки

Для разработки программного продукта были использованы следующие инструментальные и программные средства:

* IntelliJ IDEA – это интегрированная среда разработки (комплекс программных средств, которые используются для написания, исполнения, отладки и оптимизации кода) для Java, Python, Kotlin и других языков программирования от компании JetBrains. Отличается обширным наборам инструментов для рефакторинга (перепроектирования) и оптимизации кода.
* PostgreSQL — это мощная система объектно-реляционных баз данных с открытым исходным кодом. Она имеет проверенную архитектуру, которая заслужила хорошую репутацию за надежность, целостность данных и правильность. Существует в реализациях для множества UNIX-подобных платформ, а также для Microsoft Windows. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011. Сильными сторонами PostgreSQL считаются:
* высокопроизводительные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования;
* наследование;
* возможность индексирования геометрических объектов и наличие базирующегося на ней расширения PostGIS;
* встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
* расширяемость (возможность создавать новые типы данных и др.).

В процессе создания Windows-приложения «Компьютерный салон» в среде psql была разработана база данных «PkStor».

* Kotlin создавался компанией, которая делает очень много продуктов на Java и которая хорошо разбирается в современных инструментах разработки. Запрос на новый язык витает в воздухе давно, но сделать такой язык, который бы позволил взять (огромную) готовую кодовую базу Java, обычных Java-разработчиков, дать им новый инструмент и бесшовно (но более эффективно) продолжать разработку — такого инструмента до появления Kotlin не существовало.

Для разработки проекта были использованы следующие программные средства разработки:

Trello — это визуальный инструмент для управления работой, который позволяет командам обдумывать, планировать и вести совместную работу, а также отмечать успехи. Скриншот доски Trello представлен на рисунке 11

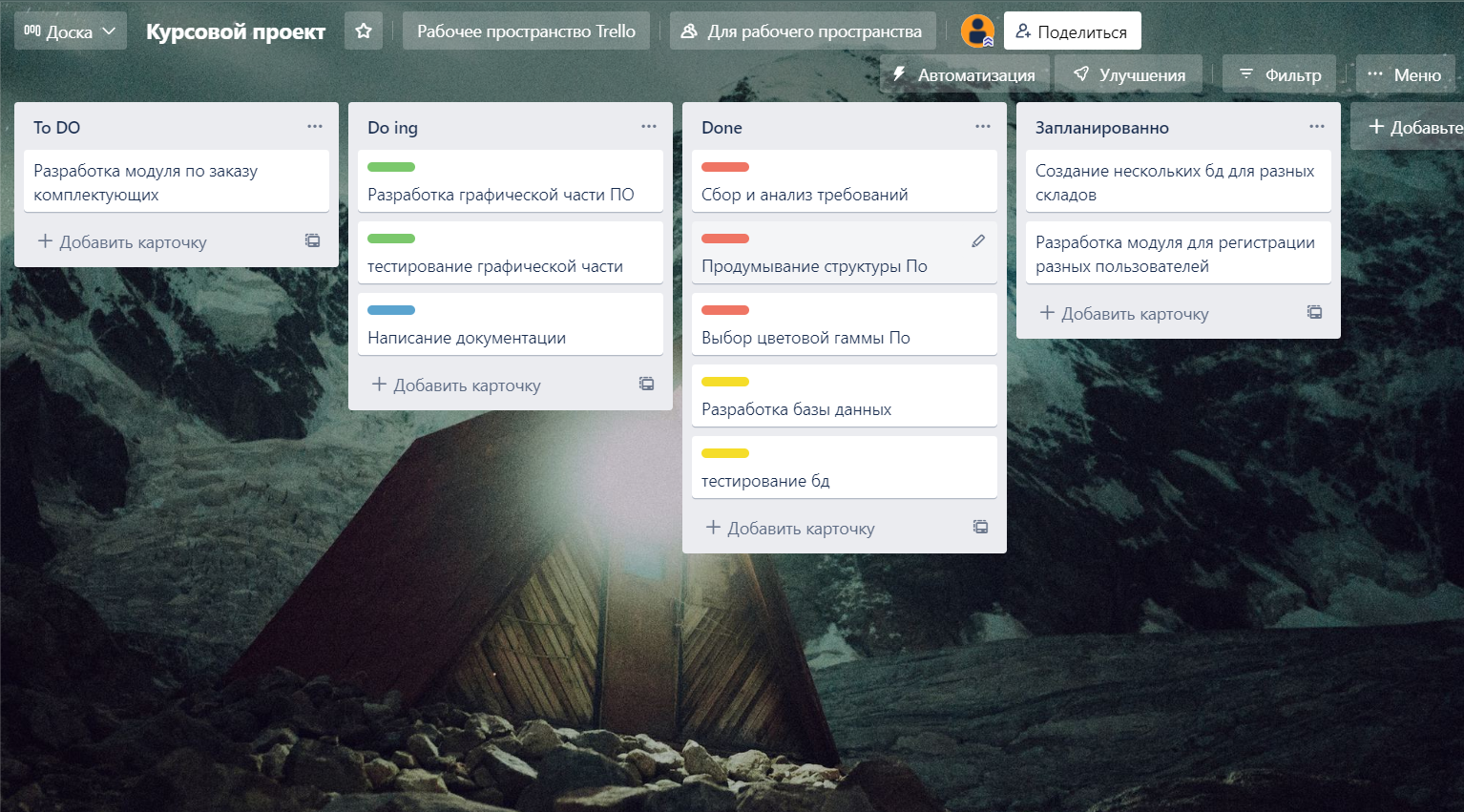


Рисунок 11

IntelliJ IDEA **—** это среда разработки, которая используется для написания, исполнения, отладки и оптимизации кода для Java, Python, kotlin. Среда разработки представлена на рисунке 12.

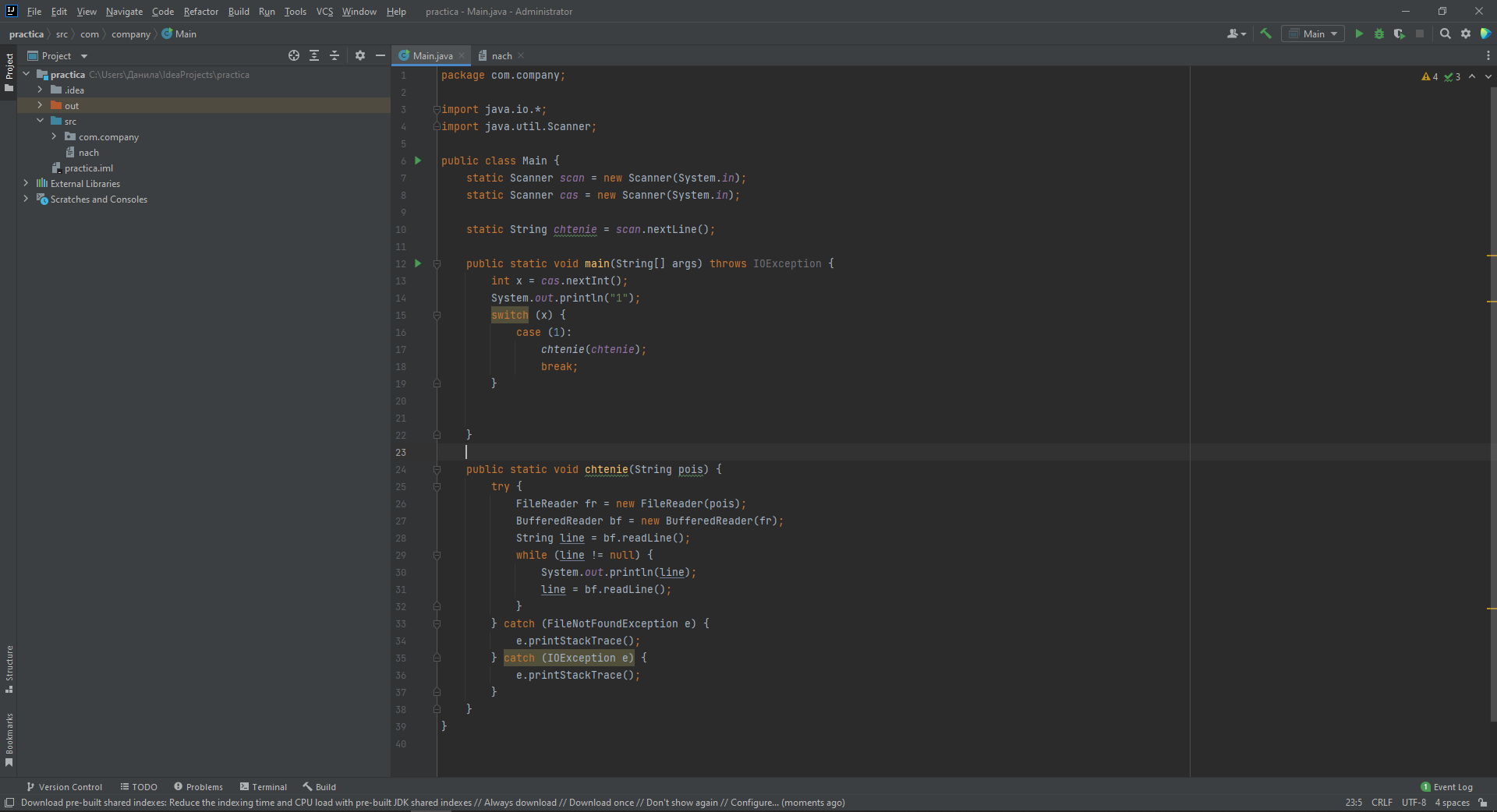


Рисунок 12

Программные коды модулей Windows-приложения «Компьютерный салон» на языке Kotlin, разработанные в среде IntelliJ IDEA, представлены в приложении А.

## 3.2 Календарный план разработки

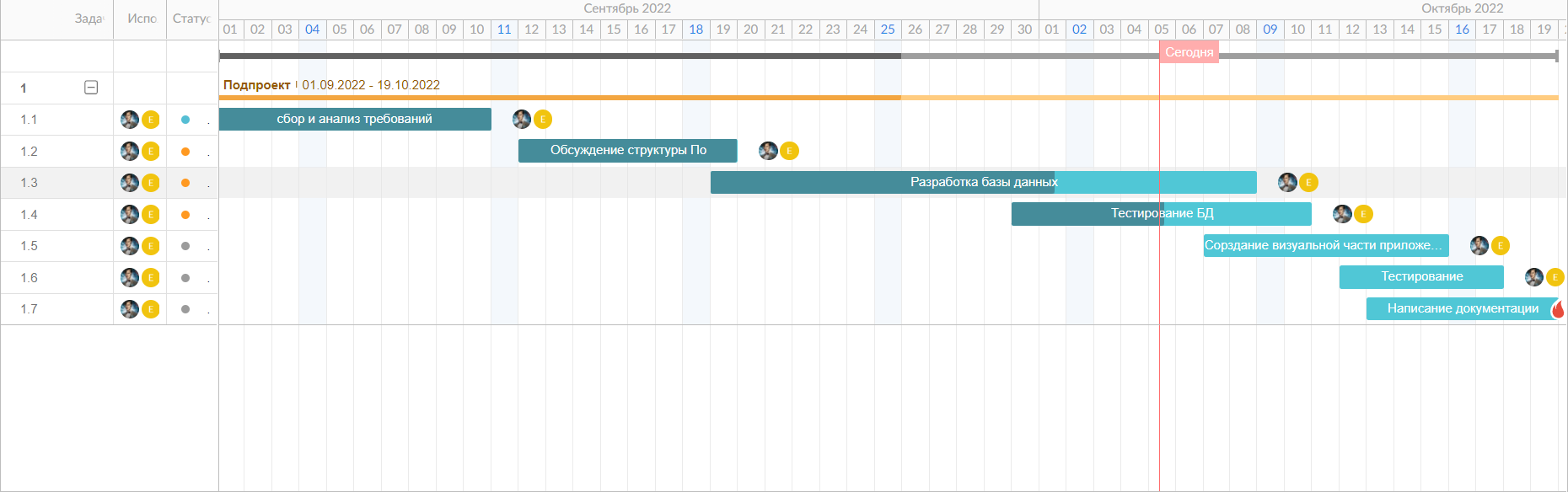
Для распределения временных сроков на разработку проекта, был создан календарный план разработки продукта с помощью диаграммы Ганта. Он представлен на рисунке 13.

Рисунок 13 – Календарный план работы

# 4 Тестирование программного продукта

## 4.1 Выбор метода обеспечения качества

Тестирование программного обеспечения ‑ это оценка разрабатываемого программного обеспечения, чтобы проверить его соответствие ожидаемым результатам. Существуют различные типы методов, используемые в области тестирования.

Тестирование – процесс контроль качества, который состоит из планирования, проектирования, собственно проверки и анализа ее результатов.

## 4.2 Тестирование программного продукта

Для проведения тестирования программного продукта был составлен план проведения тестирования, который представлен в таблице 12.

Таблица 8 – План тестирования

|  |  |  |
| --- | --- | --- |
| Тест кейс, № | Название | Описание |
| 1 | Тест авторизации пользователя Менеджер | Тест авторизации в программе под пользователем Администратор |
| 2 | Тест авторизации пользователя Продавец | Тест авторизации в программе под пользователем Продавец |
| 3 | Тест занесения новых комплектующих в базу данных | Тест возможности сохранения данных новых комплектующих в базе данных |
| 4 | Тест открытия списка товаров | Тест загрузки информации из базы данных в табличный компонент формы приложения |

Тест-кейсы для проведения тестирования представлены в таблицах 13-16.

Таблица 13 – Тест-кейс 1

|  |  |  |
| --- | --- | --- |
| Идентификатор | | 1 |
| Название | | Тест авторизации пользователя Администратор |
| Приоритет | Высокий | |
| Описание | Тестируется правильность соединения с базой данных и перехода на главную форму приложения | |
| Предусловие | Предусловие отсутствует | |
| Шаги тестирования | 1. Запустить проект (форму авторизации) 2. Ввести нужный логин в окно пользователя 3. Ввести пароль 4. Нажать на кнопку «Войти» | |
| Ожидаемый результат | При корректном вводе данных: переход на главную форму приложения с полностью доступным функционалом;  при некорректном вводе данных: сообщение об ошибке | |
| Постусловие | Постусловие отсутствует | |
| Фактический результат | При корректном вводе данных: происходит переход на главную форму приложения с полностью доступным функционалом;  при некорректном вводе данных: отображается сообщение об ошибке | |
| Статус | Пройден | |

Таблица 14 – Тест-кейс 2

|  |  |
| --- | --- |
| Идентификатор | 2 |
| Название | Тест авторизации пользователя Продавец |
| Приоритет | Высокий |
| Описание | Тестируется правильность соединения с базой данных и перехода на главную форму приложения |
| Предусловие | Предусловие отсутствует |
| Шаги тестирования | 1. Запустить проект (форму авторизации) 2. Ввести нужный логин в окне пользователя 3. Ввести пароль 4. Нажать на кнопку «Войти» |
| Ожидаемый результат | При корректном вводе данных: переход на главную форму приложения с возможностью добавления товара в базу данных, составление наряда, поиск комплектующих, просмотр полного списка комплектующих;  при некорректном вводе данных: сообщение об ошибке |

Продолжение таблицы 14

|  |  |
| --- | --- |
| Постусловие | Постусловие отсутствует |
| Фактический результат | При корректном вводе данных: переход на главную форму приложения с возможностью добавления товара в базу данных, составление наряда, поиск комплектующих, просмотр полного списка комплектующих;  при некорректном вводе данных: отображается сообщение об ошибке |
| Статус | Пройден |

Таблица 15 – Тест-кейс 3

|  |  |
| --- | --- |
| Идентификатор | 3 |
| Название | Тест занесения новых комплектующих в базу данных |
| Приоритет | Высокий |
| Описание | Тестируется правильность внесения данных в БД при использовании приложения |
| Предусловие | Предусловие отсутствует |
| Шаги тестирования | 1. Авторизоваться в приложении в роли пользователя Менеджера или продавца 2. Открыть форму справочника комплектующих с выбором пункта «комплектующие» из меню «Данные» главной формы приложения 3. Ввести данные нового комплектующего в очищенные текстовые поля формы 4. Нажать кнопку «Сохранить данные» |
| Ожидаемый результат | При корректном вводе данных: товар добавляется в базу данных, и менеджер или продавец может просмотреть информацию об этом товаре, продать его;  при некорректном вводе данных: сообщение об ошибке |
| Постусловие | Постусловие отсутствует |
| Фактический результат | При некорректных данных сообщение об ошибке не отображается |
| Статус | Провален |

Таблица 16 – Тест-кейс 4

|  |  |
| --- | --- |
| Идентификатор | 4 |
| Название | Тест открытия списка товаров |
| Приоритет | Высокий |
| Описание | Тестируется правильность загрузки информации из базы данных в табличный компонент формы при использовании приложения |
| Предусловие | Предусловие отсутствует |
| Шаги тестирования | 1. Авторизоваться в приложении в роли пользователя Продавец 2. Открыть форму списка товаров |
| Ожидаемый результат | На форме списка товаров отображение таблицы из 5 столбцов («Номер», «Название», «Стоимость», «Количество», «Наличие»); ширина столбцов таблицы скорректирована под их содержимое; нет выделенных (активных) ячеек таблицы; |
| Постусловие | Постусловие отсутствует |
| Фактический результат | При открытии формы со списком товаров - выводится полный список товаров в салоне в нужных столбцах («Номер», «Название», «Стоимость», «Количество», «Наличие»). |
| Статус | Пройден |

# 5 Разработка документации на программный продукт

К видам документации, разрабатываемой на разных этапах жизненного цикла программного продукта, относятся:

1. технические требования;
2. руководства (например, по установке ПП, пользователя, администратора, программиста, по технической поддержке и т.д.);
3. описание проекта;
4. задания исполнителям (задание, распределённое между конкретными людьми или группами, участвующими в реализации проекта);
5. отчёты о ходе работ ‑ создаются менеджерами для контролирующих органов;

Эксплуатационные программные документы (документы, входящие в состав программных систем) ‑ документы, предназначенные для конечных пользователей или же обслуживающего персонала, позволяющие им осваивать и квалифицированно применять эти системы для решения конкретных функциональных задач.

# Заключение

В данный курсовой работе мы изучили различные технологии анализа требований и проектирования программного обеспечения, а также выполняя разработку Windows-приложения, мы смогли воспользоваться этими технологиями. Рассмотрели возможности реализации и тестирования компьютерных систем.

# Список литературы

1. <https://kotlinlang.org/> (Официальный сайт kotlin)
2. <https://metanit.com/kotlin/tutorial/> (Справочник по Kotlin)
3. <https://tproger.ru/articles/creating-an-app-for-kotlin-multiplatform/> (Справочник по созданию графической части на Kotlin)
4. <https://tproger.ru/translations/kotlin-vs-java-android/> (Руководство по Kotlin)
5. <https://www.youtube.com/channel/UCP7uiEZIqci43m22KDl0sNw> (Официальный ютуб канал компании JetBrains по языку Kotlin)

Приложение А

A.1.

import DataBase.DemonstrateProducts

import DataBase.Program

import DataBase.WorkWithDb

fun main(args: Array<String>) {

val x = Program()

x.work()

}

A.2.

package DataBase

import java.sql.DriverManager.println

class Program {

val workWithDataBAse = WorkWithDb()

val demonstrateProducts = DemonstrateProducts()

fun work() {

while (true) {

println(

"Выбирите действие 1-добавить продукт, 2 обновить информацию о продукте\n" +

"3 - посмотреть стоймость товара ,4 - посмотреть информацию о продукте, 5 - помостреть базу данных, 6 - выйти"

)

when (readln().toInt()) {

1 -> {

println("Введите колличество продуктов")

val countProduct = readln().toInt()

println("Введите название продукта ")

for (i in 1..countProduct) {

val name = readln()

println("Введите колличество продукта на складе ")

val count = readln().toInt()

println("Введите стоймость продукта ")

val price = readln().toInt()

workWithDataBAse.addProduct(name, count, price)

}

}

2 -> {

println("Введите id продукта ")

val id = readln().toInt()

println("Введите название продукта ")

val name = readln()

println("Введите колличество продуктов ")

val count = readln().toInt()

println("Введите стоймость продукта ")

val price = readln().toInt()

workWithDataBAse.editProduct(id, name, count)

}

3 -> {

println("Введите название продукта ")

val name = readln()

println("Введите колличество продуктов ")

val count = readln().toInt()

demonstrateProducts.calculatePrice(name, count)

}

4 -> {

println("Введите название продукта ")

val name = readln()

demonstrateProducts.showProduct(name)

}

5 -> {

demonstrateProducts.showAllProducts()

}

6 -> break;

}

}

}

}

A.3.

package DataBase

import org.jetbrains.exposed.sql.\*

import org.jetbrains.exposed.sql.transactions.transaction

class DemonstrateProducts {

val connect = Database.connect(

"jdbc:postgresql://localhost:5432/Shop", driver = "org.postgresql.Driver",

user = "postgres", password = "postgres"

)

fun showAllProducts() {

for (product in transaction { Product.selectAll().toList()}){

println(product.toString().replace("DataBase.Product.",""))

}

}

fun showProduct(name: String){

val products = transaction { Product.select { Product.productName eq name }.toList() }

for(product in products){

println(product.toString().replace("DataBase.Product.",""))

}

}

fun calculatePrice(name: String, count: Int){

val products = transaction { Product.slice(Product.ProductPriceForOne).select { Product.productName eq name }.toList() }

for(product in products){

println("$name цена за за колличество = ${product[Product.ProductPriceForOne] \* count}")

}

}

}

A.4.

package DataBase

import org.jetbrains.exposed.dao.id.IntIdTable

object Product: IntIdTable() {

val productName = varchar("name",100)

var productCount = integer("count")

var ProductPriceForOne = integer("priceForOne")

}

A.5.

package DataBase

data class ProductInfo(var productName: String, var productCount: Int, var productPrice: Int)

A.6.

package DataBase

import org.jetbrains.exposed.sql.\*

import org.jetbrains.exposed.sql.transactions.transaction

class WorkWithDb {

val connect = Database.connect("jdbc:postgresql://localhost:5432/Shop", driver = "org.postgresql.Driver",

user = "postgres", password = "postgres")

init {

transaction {

SchemaUtils.create(Product)

}

}

fun addProduct(productName: String, productCount: Int, productPrice: Int){

transaction {

Product.insert {

it[this.productName] = productName

it[this.productCount] = productCount

it[ProductPriceForOne] = productPrice

}

}

}

fun editProduct(id: Int, name: String = "", count: Int = 0, priceForOne: Int = 0){

val productInfo = getProduct(id)

val product = ProductInfo(productInfo[Product.productName], productInfo[Product.productCount],productInfo[Product.ProductPriceForOne])

if (name != "") {

product.productName = name

}

else if (count != null){

product.productCount = count

}

else {

product.productPrice = priceForOne

}

print("$product\nХотите сохранить информацию ")

val isEdit = when(readln()){

"Да" -> true

else -> false

}

if(isEdit){

edit(id, product)

}

}

private fun getProduct(id: Int) = transaction {

Product.select {

Product.id eq id

}.firstOrNull()?: throw Exception("Id doe's not exist")

}

private fun edit(id: Int, product: ProductInfo){

transaction {

Product.update({ Product.id eq id }) {

with(SqlExpressionBuilder) {

it[productName] = product.productName

it[productCount] = product.productCount

it[ProductPriceForOne] = product.productPrice

}

}

}

}

}