

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Факультет прикладной математики и информатики

Кафедра технологий программирования

Жуковский Павел Сергеевич

«Генерация кода проектируемого программного обеспечения и разработка приложения. Тестирование разработанного приложения и финальный отчет»

Отчет по лабораторной работе №6  
«Проектирование программных систем» студента 2 курса 13 группы  
Вариант 10

**Преподаватель  
Давидовская Мария Ивановна**

**Минск 2019**

## Оглавление

<b>1. Описание предметной области .....</b>	<b>4</b>
<b>2. Постановка задачи .....</b>	<b>5</b>
<b>3. Определение требований .....</b>	<b>6</b>
<b>3.1. Требования к системе .....</b>	<b>6</b>
<b>3.2. Глоссарий .....</b>	<b>6</b>
<b>3.3. Функциональные возможности .....</b>	<b>7</b>
<b>3.4. Требования по реализации .....</b>	<b>7</b>
<b>3.5. Надёжность.....</b>	<b>7</b>
<b>3.6. Производительность.....</b>	<b>7</b>
<b>3.7. Безопасность .....</b>	<b>7</b>
<b>4. Анализ вариантов использования.....</b>	<b>8</b>
<b>4.1. Основная диаграмма.....</b>	<b>8</b>
<b>4.2. Описание актеров .....</b>	<b>8</b>
<b>4.3. Варианты использования.....</b>	<b>9</b>
<b>4.4. Спецификации.....</b>	<b>10</b>
<b>4.5. Бизнес-диаграмма «Business diagram».....</b>	<b>15</b>
<b>4.6. Дополнительная диаграмма прецедентов «Making order» .....</b>	<b>16</b>
<b>4.7. Дополнительная диаграмма прецедентов «Signing in» .....</b>	<b>17</b>
<b>4.8. Дополнительная диаграмма претендентов «Checking Order out»....</b>	<b>18</b>
<b>4.9. Диаграммы деятельности .....</b>	<b>19</b>
<b>4.10. Диаграммы состояний .....</b>	<b>22</b>
<b>4.11. Диаграммы классов.....</b>	<b>23</b>
<b>4.12. Диаграммы последовательностей .....</b>	<b>28</b>
<b>4.13. Диаграмма объектов .....</b>	<b>33</b>
<b>5. Проектирование архитектуры и элементов системы.....</b>	<b>34</b>
<b>5.1. Диаграмма пакетов .....</b>	<b>34</b>
<b>5.2. Диаграмма развёртывания.....</b>	<b>34</b>

5.3. Первоначальные представления макета интерфейса .....	35
5.4. Модель базы данных .....	37
5.5. Диаграмма компонентов .....	38
6. Разработка приложения.....	39
6.1. Сгенерированные классы Modelio .....	39
6.2. Исходный код приложения .....	45
6.3. Скриншоты работы приложения .....	60
6.4. Ссылка на все исходные файлы проекта (включая sql-файл) .....	62
7. Результаты тестирования системы .....	63
7.1. Unit testing.....	63
7.2. Integration testing.....	64
7.3. System testing .....	64
7.4. Usability testing.....	65
7.5. Acceptance testing.....	65
7.6. Scenario testing .....	66

## **1. Описание предметной области**

### **Вариант 10. АИС «Обслуживание заказов клиентов предприятия»**

#### Описание предметной области

Предприятие (Код, Название, Краткое название) осуществляет доставку разных товаров (Код, Название, Краткое название) населению. Прием заказов от населения осуществляет специальная служба (Код, Название, Краткое название) предприятия.

Для того чтобы стать потребителем услуг предприятия каждый абонент должен зарегистрироваться, при этом фиксируются его ФИО, адрес, телефон и паспортные данные (Серия, Номер, Дата выдачи, Кем выдан). Каждый абонент в течение дня может сделать несколько заказов (Дата, Время), заказу присваивается номер.

В каждом заказе может содержаться несколько наименований товаров, для каждого указывается количество, единица измерения (Код, Название, Краткое Название), цена за единицу товара, общая стоимость товара. Заказ также имеет итоговую сумму.

При формировании бланка заказа, который будет подписан абонентом при получении товара, фиксируется, оплачен заказ, или абонент получает товар в кредит. Также на бланке заказа указывается: реквизиты предприятия (название, адрес, контактные телефоны); ФИО и должность оператора, принявшего заказ; ФИО, должность сотрудника, доставившего заказ.

Необходимо осуществлять следующую обработку данных:

- список товаров (код, наименование), пользующихся наибольшим спросом (максимальное количество позиций заказов) у населения за заданный период;
- динамика изменения стоимости заданного товара за заданный период по месяцам;
- список наименований улиц, на которых проживают абоненты предприятия по убыванию числа абонентов.

## **2. Постановка задачи**

Система создаётся с целью упрощения и систематизации взаимодействия клиентов предприятия с поставщиками товаров, а также для сохранения данных о получении клиентами того или иного заказа.

### 3. Определение требований

#### 3.1. Требования к системе

Необходимо хранить информацию об организации, помещениях, договорах и самих клиентах. Обеспечить возможность регистрации для клиентов в системе. Обеспечить возможность клиенту сделать несколько заказов, а также возможность уточнить детали каждого из заказов (количество, цена и т.д.). Обеспечить хранение информации о состоянии заказов клиентов, а также данных об этих заказах. Создать статистику относительно выполненных заказов (список товаров, пользующихся наибольшим спросом у населения за заданный период; динамика изменения стоимости заданного товара за заданный период по месяцам; список наименований улиц, на которых проживают абоненты предприятия по убыванию числа абонентов).

#### 3.2. Глоссарий

Предприятие (Company)	Система, обеспечивающая доставку разных товаров населению.
Специальная служба (Special service)	Организация предприятия, осуществляющая приём заказов от населения.
Потребитель услуг (Services' consumer)	Абонент, зарегистрировавшийся в системе.
Абонент (Subscriber)	Клиент, оформляющий заказ/заказы в системе.
Заказ (Order)	Совокупность некоторых наименований товаров.
Бланк заказа (Order form)	Документ, содержащий информацию о заказе.
Реквизиты предприятия	Контакты предприятия

	(название, адрес, телефоны).
Оператор	Человек, принимающий заказы в системе.

### **3.3. Функциональные возможности**

Система должна уметь: 1) составлять список товаров, пользующихся наибольшим спросом у населения за заданный период; 2) воспроизводить динамику изменения стоимости заданного товара за заданный период по месяцам; 3) составлять список наименований улиц, на которых проживают абоненты предприятия по убыванию числа абонентов.

### **3.4. Требования по реализации**

Система должна быть совместима с операционной системой Windows.

### **3.5. Надёжность**

Система должна быть в работоспособном состоянии 24 часа в день, 7 дней в неделю.

### **3.6. Производительность**

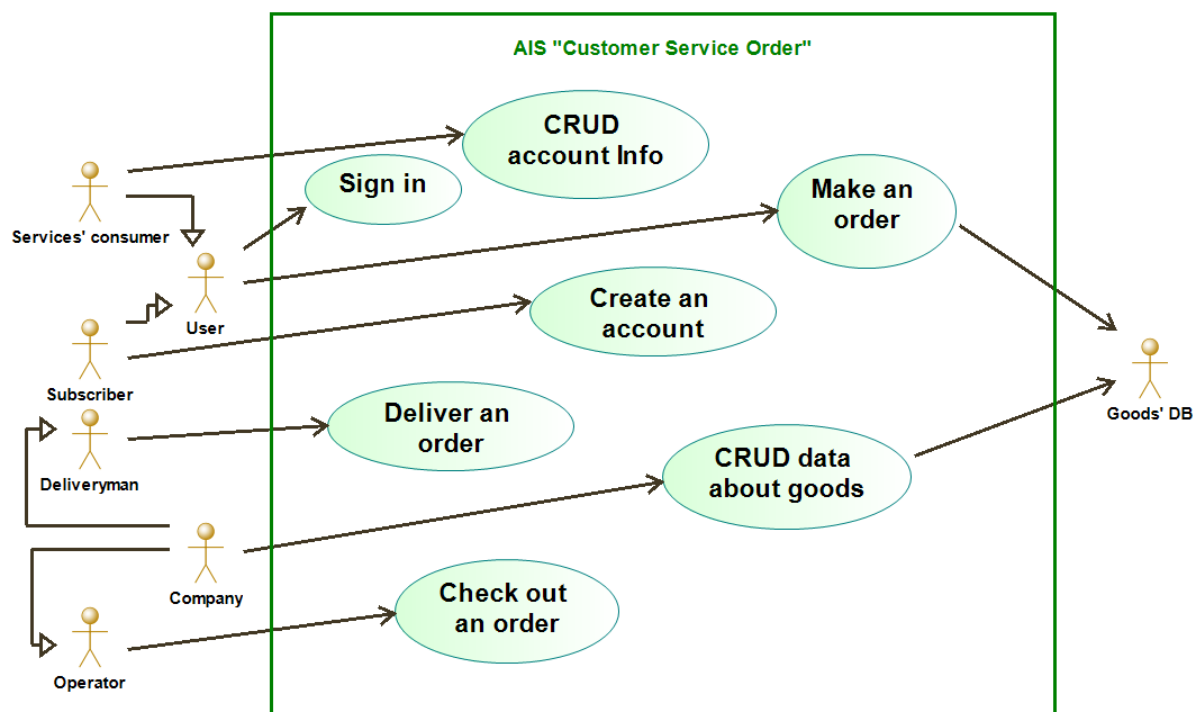
Система должна поддерживать несколько тысяч одновременно работающих пользователей (эти требования могут изменяться в ходе роста популярности системы).

### **3.7. Безопасность**

В системе должна быть предусмотрена защита личной информации о заказах потребителей услуг, а также вывод необходимой информации о платеже клиентов.

## 4. Анализ вариантов использования

### 4.1. Основная диаграмма



### 4.2. Описание актеров

Потребитель услуг (Services' consumer)	Абонент, зарегистрировавшийся в системе
Абонент (Subscriber)	Клиент, работающий с системой
Пользователь (User)	Человек, работающий с системой (включает в себя как потребителей услуг, так и абонентов)
Доставщик (Deliveryman)	Человек из специальной службы, обеспечивающий доставку товаров абонентам.
Предприятие (Company)	Организация, отвечающая за работу всей системы



Оператор (Operator)	Человек из предприятия, обеспечивающий оформление заказов абонентов
БД с товарами (Goods' DB)	База данных, в которой хранится информация обо всех товарах

#### 4.3. Варианты использования

СЧУУ информацию об аккаунте (CRUD account Info)	Создать/Читать/Улучшить/Удалить какую-то информацию об аккаунте (используется потребителями услуг)
Войти (Sign in)	Войти в систему с товарами для дальнейшего их рассмотрения и покупки (используется пользователями)
Сделать заказ (Make an order)	Возможность для пользователя совершить заказ с целью покупки того или иного товара
Создать аккаунт (Create an account)	Возможность для абонента создать свой аккаунт в системе для удобства дальнейших покупок товаров
Доставить заказ (Deliver an order)	Доставить заказ с теми или иными товарами пользователю системы (данная функция используется доставщиком)
СЧУУ данные о товарах (CRUD data about goods)	Создать/Читать/Улучшить/Удалить данные о тех или иных товарах в системе (основная функция предприятия)
Обработать заказ (Check out an order)	Оформить заказ того или иного пользователя системы (функция оператора)

## 4.4. Спецификации

### *1. Сценарий событий для прецедента «СЧУУ информацию об аккаунте».*

#### 1.1 Предусловия

Для пользователей, которые зарегистрировались в системе для дальнейшего удобства в покупке тех или иных товаров, нужна возможность всяческого редактирования информации об их аккаунтах.

#### 1.2 Главный сценарий

Происходит одно из следующих четырёх действий: 1) Потребитель услуг создаёт какую-то информацию об аккаунте на стене (В-1); 2) Потребитель услуг читает какую-то информацию со своего аккаунта (В-2), если она имеется (А-1); 3) Потребитель услуг улучшает, или добавляет/совершенствует, какую-то информацию в своём аккаунте (В-3), если она имеется (А-1); 4) Пользователь может также удалить какую-то информацию с аккаунта (В-4), если она имеется (А-1).

#### 1.3 Альтернативные сценарии

А-1: Нету никакой информации об аккаунте, чтобы ее читать/улучшать/удалять. Пользователю необходимо сначала создать информацию (В-1).

#### 1.4 Дополнительные сценарии

В-1: Создание информации на аккаунте.

Пользователь создает информацию на аккаунте.

В-2: Чтение информации на аккаунте.

Пользователь читает какую-то информацию на своём аккаунте.

В-3: Улучшение информации на аккаунте.

Пользователь улучшает, или добавляет/уточняет, какую-то информацию на своём аккаунте.

В-4: Удаление информации на аккаунте.

Пользователь в праве также удалить информацию на своём аккаунте.

#### 1.5 Постусловия

В информацию на аккаунте вносятся правки.

### *2. Сценарий событий для прецедента «Войти».*

#### 2.1 Предусловия

Для любого взаимодействия с системой с товарами пользователям необходимо в неё войти. Для этого не обязательно иметь созданный аккаунт в системе, хотя желательно, например, чтобы следить за аналитикой истории своих покупок.

## 2.2 Главный сценарий

Вариант использования начинает выполняться, когда пользователь системы подключается к системе, вводит свое имя и пароль. Система проверяет правильность пароля (А-1). Если пароль не верен, пользователю предлагается сменить пароль (А-2) или выйти из системы (А-3).

## 2.3 Альтернативные сценарии

А-1: введено неправильное имя или пароль.

Пользователь должен повторить ввод или завершить прецедент.

А-2: если пользователь 3 раза ввёл неверный пароль, ему предлагается сменить пароль с помощью электронной почты.

А-3: пользователь может выйти из системы.

## 2.4 Постусловия

Если вход выполнен успешно, пользователь получает доступ к системе.

# *3. Сценарий событий для прецедента «Сделать заказ».*

## 3.1 Предусловия

Сделать заказ – основная функция, которая разрабатывается системой для пользователя. Чтобы совершить покупку того или иного товара, пользователю необходимо этот заказ сделать.

## 3.2 Главный сценарий

Пользователь выбирает какой-то интересующий его товар, может выбрать то, какое количество этого товара он хочет заказать (В-1), а также указывает свою контактную информацию для последующего общения с оператором и заполнения бланка заказа.

## 3.3 Дополнительные сценарии

В-1: Выбор определенного количества товара.

Если пользователю системе требуется более одного товара, то он задает какое-то определённое количество товара, которое ему нужно.

## 3.4 Постусловия

После совершения заказа, происходит его оформление оператором.

# *4. Сценарий событий для прецедента «Создать аккаунт».*

## 4.1 Предусловия

Для незарегистрированных пользователей (абонентов) нужно реализовать возможность создать аккаунт. Это упростит их взаимодействие с системой и наделит их дополнительными возможностями.

#### 4.2 Главный сценарий

Пользователь вводит имя, и, если оно удовлетворяет требованиям для имён аккаунтов (А-1), то он вводит пароль, и, если он удовлетворяет требованиям для паролей (А-2), создает аккаунт с заданным именем и паролем. Пользователь также может привязать аккаунт к мобильному телефону (В-1) или к своей электронной почте (В-2).

#### 4.3 Альтернативные сценарии

А-1: Введённое имя аккаунта не удовлетворяет требованиям.

Введённое имя либо слишком длинное, либо слишком короткое, либо содержит недопустимые символы, либо оно уже занято. Требуется ввести другое имя или прекратить попытку регистрации.

А-2: Введённый пароль не удовлетворяет требованиям.

Введённый пароль либо слишком короткий, либо слишком длинный, либо слишком простой. Требуется ввести другой пароль или прекратить попытку регистрации.

#### 4.4 Дополнительные сценарии

В-1: Привязка аккаунта к мобильному телефону.

Пользователь может выслать на свой мобильный номер сообщение с подтверждением, после перехода по которому аккаунт привязывается к мобильному телефону, обеспечивая большую безопасность аккаунта.

В-2: Привязка аккаунта к электронной почте.

Пользователь может выслать на свою электронную почту сообщение с подтверждением, после перехода по которому аккаунт привязывается к электронной почте, обеспечивая большую безопасность аккаунта.

#### 4.5 Постусловия

После успешного выбора имени аккаунта и пароля регистрируется аккаунт для пользователя системы.

### *5. Сценарий событий для прецедента «Доставить заказ».*

#### 5.1 Предусловия

Для того, чтобы заказ был доставлен пользователю системы, требуется сделать функцию доставки товара доставщиком до того или иного заказчика.

## 5.2 Главный сценарий

Доставщик доставляет заказ пользователю в соответствии с информацией об адресе пользователя, выданному ему специальной службой предприятия. После этого он успешно доставляет заказ, если после прибытия по указанному адресу пользователь системы был на месте (А-1) и забрал заказ (А-2).

## 5.3 Альтернативные сценарии

А-1: Пользователя системы нет на месте.

Если по каким-то причинам пользователя системы, сделавшего заказ, по указанному адресу не оказалось, то доставщику следует прийти в другой раз, либо позвонить своему заказчику для уточнения дальнейших действий. Если пользователь недоступен, то следует обратиться к предприятию.

А-2: Пользователь системы не принял заказ.

Если по каким-то причинам пользователь систему отказался принимать заказ, то доставщику следует обратиться к своему предприятию для указания дальнейших действий.

## 5.4 Постусловия

После успешной операции заказ доставляется пользователю системы.

## *6. Сценарий событий для прецедента «СЧУУ данные о товарах».*

### 6.1 Предусловия

Всяческое редактирование базы данных с товарами есть основная работа предприятия. Для того, чтобы вносить или убирать какие-то изменения в системе, предприятию нужна данная функция.

### 6.2 Главный сценарий

Происходит одно из следующих четырёх действий: 1) Работник предприятия создаёт какую-то информацию о товаре в базе данных (В-1); 2) Работник предприятия читает какую-то информацию о товаре в базе данных (В-2), если она имеется (А-1); 3) Работник предприятия улучшает, или добавляет/совершенствует, какую-то информацию о том или ином товаре (В-3), если она имеется (А-1); 4) Работник предприятия может также удалить какую-то информацию о товаре с базы данных (В-4), если она имеется (А-1).

### 6.3 Альтернативные сценарии

А-1: Нету никакой информации о товаре, чтобы ее читать/улучшать/удалять. Работнику предприятия необходимо сначала создать информацию (В-1).

### 6.4 Дополнительные сценарии

В-1: Создание информации о товаре в базе данных.

Работник предприятия создает информацию о товаре в базе данных.

В-2: Чтение информации о товаре с базы данных.

Работник предприятия читает какую-то информацию о товаре с базы данных.

В-3: Улучшение информации о товаре в базе данных.

Работник предприятия улучшает, или добавляет/уточняет, какую-то информацию о товаре в базе данных

В-4: Удаление информации о товаре с базы данных.

Работник предприятия в праве также удалить информацию о товаре с базы данных.

### 6.5 Постусловия

В информацию о товаре, располагающемся в базе данных товаров, вносятся правки.

## *7. Сценарий событий для прецедента «Обработать заказ».*

### 7.1 Предусловия

Как только пользователь системы совершает заказ, оператор должен этот заказ обработать и оформить. Для этого оператору потребуется эта функция.

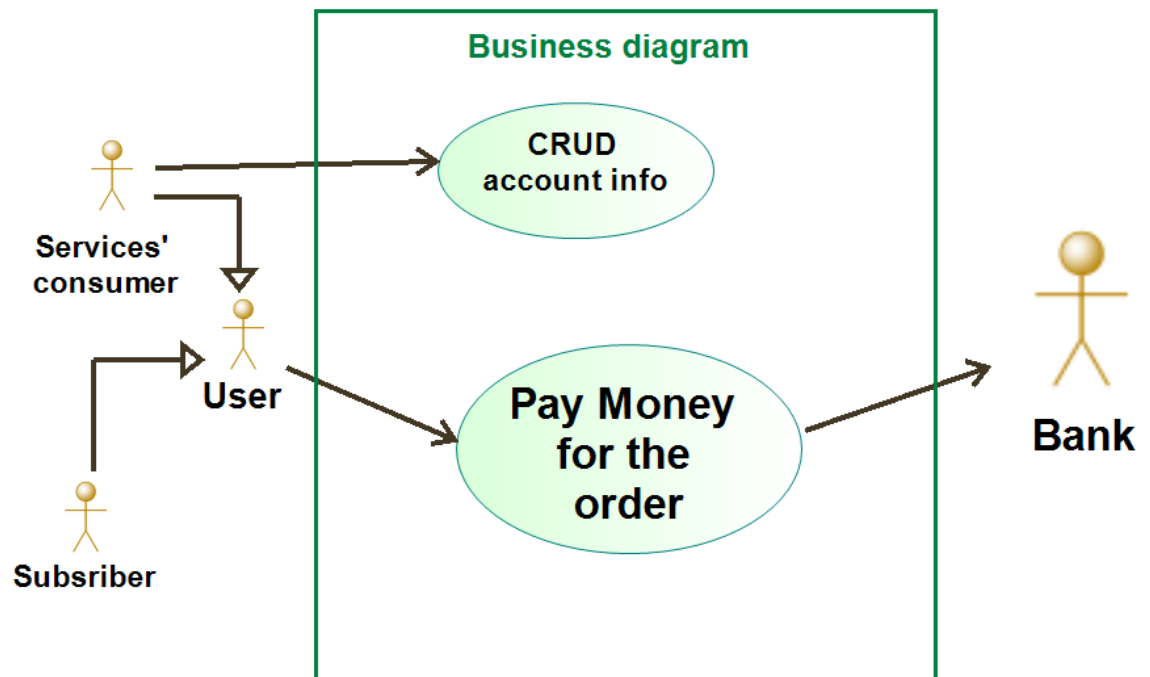
### 7.2 Главный сценарий

Оператор, работающий на предприятие обрабатывает и оформляет заказ пользователя системы, также помогая тому составлять бланк заказа. Он также собирает всю информацию о заказе, контактную информацию заказчика, после чего передаёт ее доставщику.

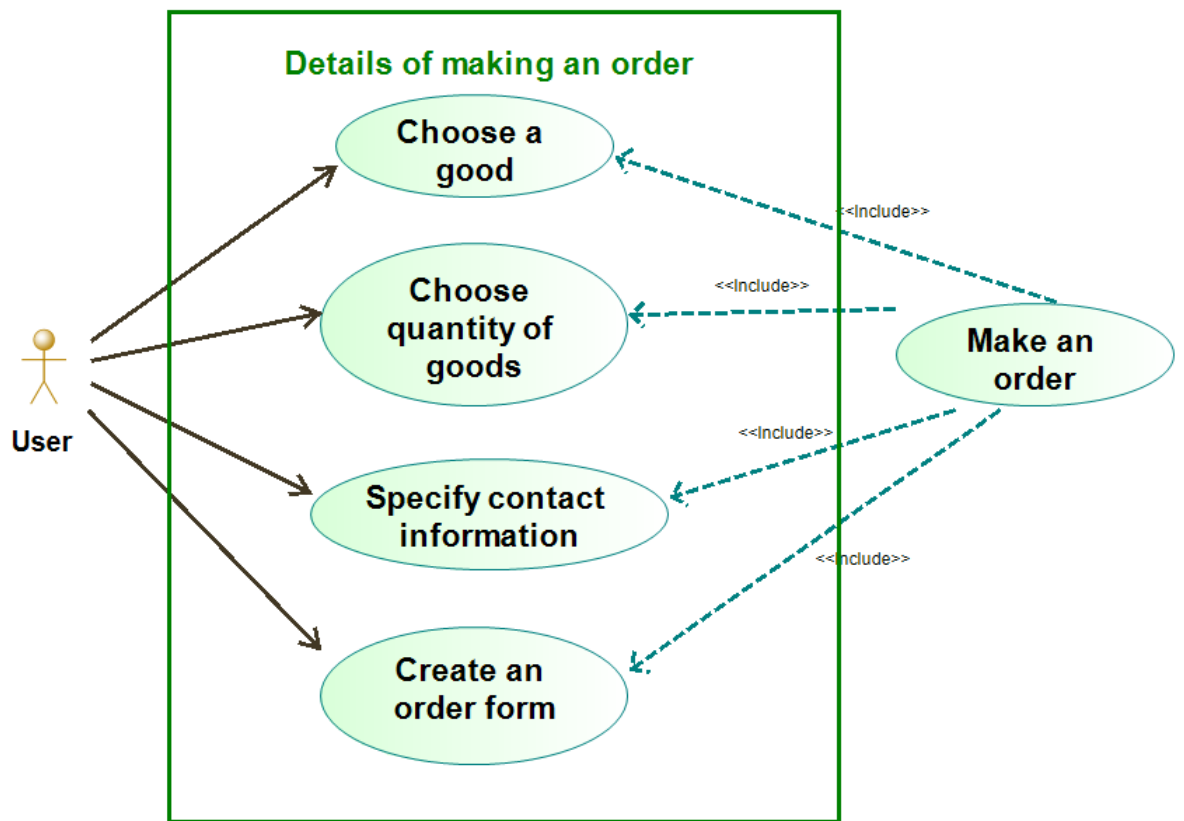
### 7.3 Постусловия

Происходит оформление и уточнение деталей заказа.

#### 4.5. Бизнес-диаграмма «Business diagram»

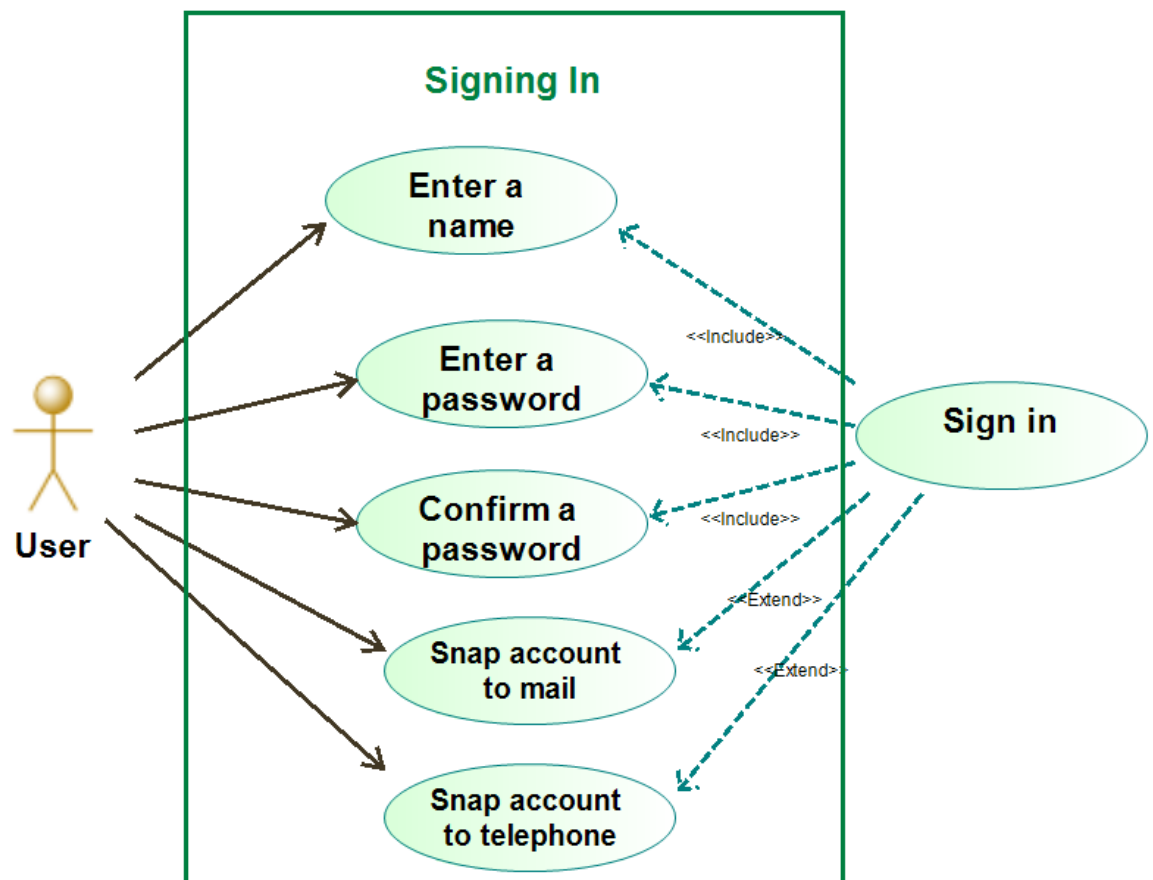


#### 4.6. Дополнительная диаграмма прецедентов «Making order»

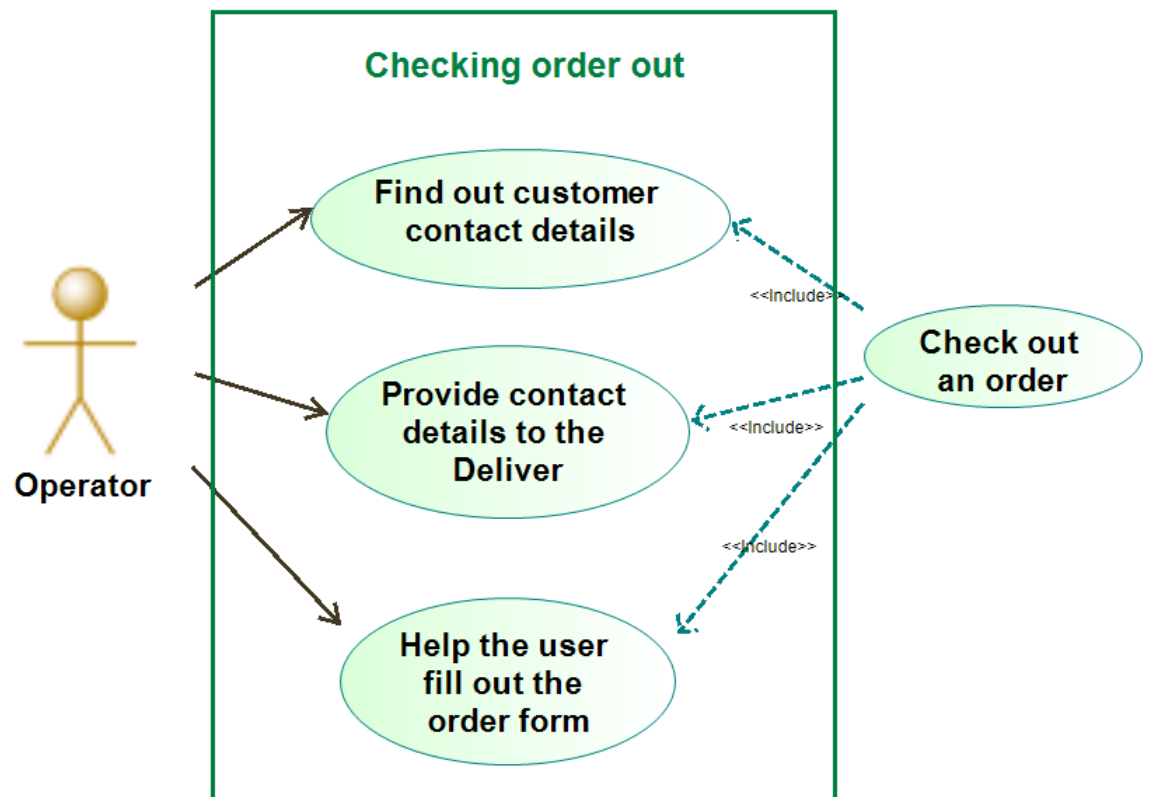




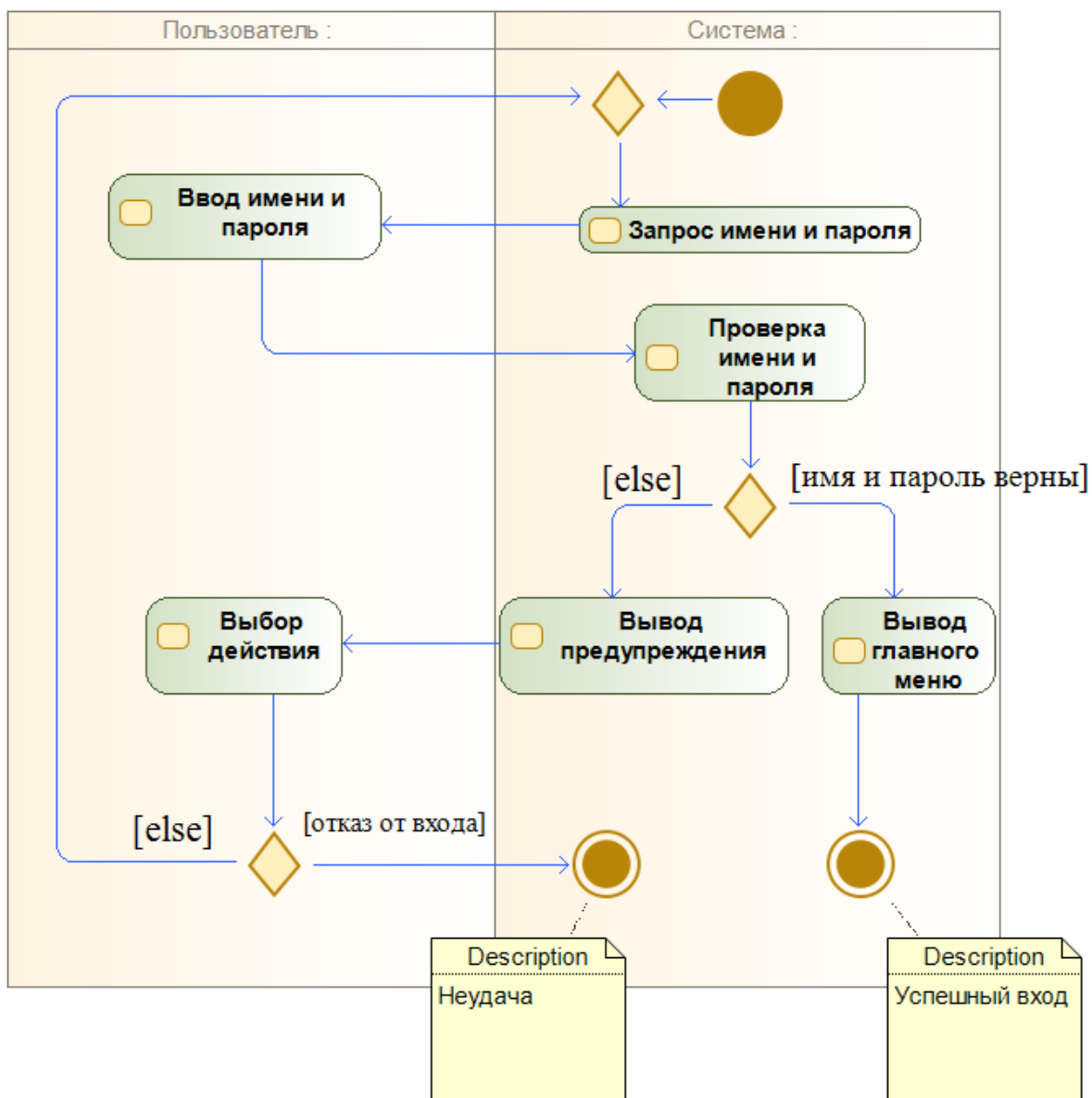
#### 4.7. Дополнительная диаграмма прецедентов «Signing in»



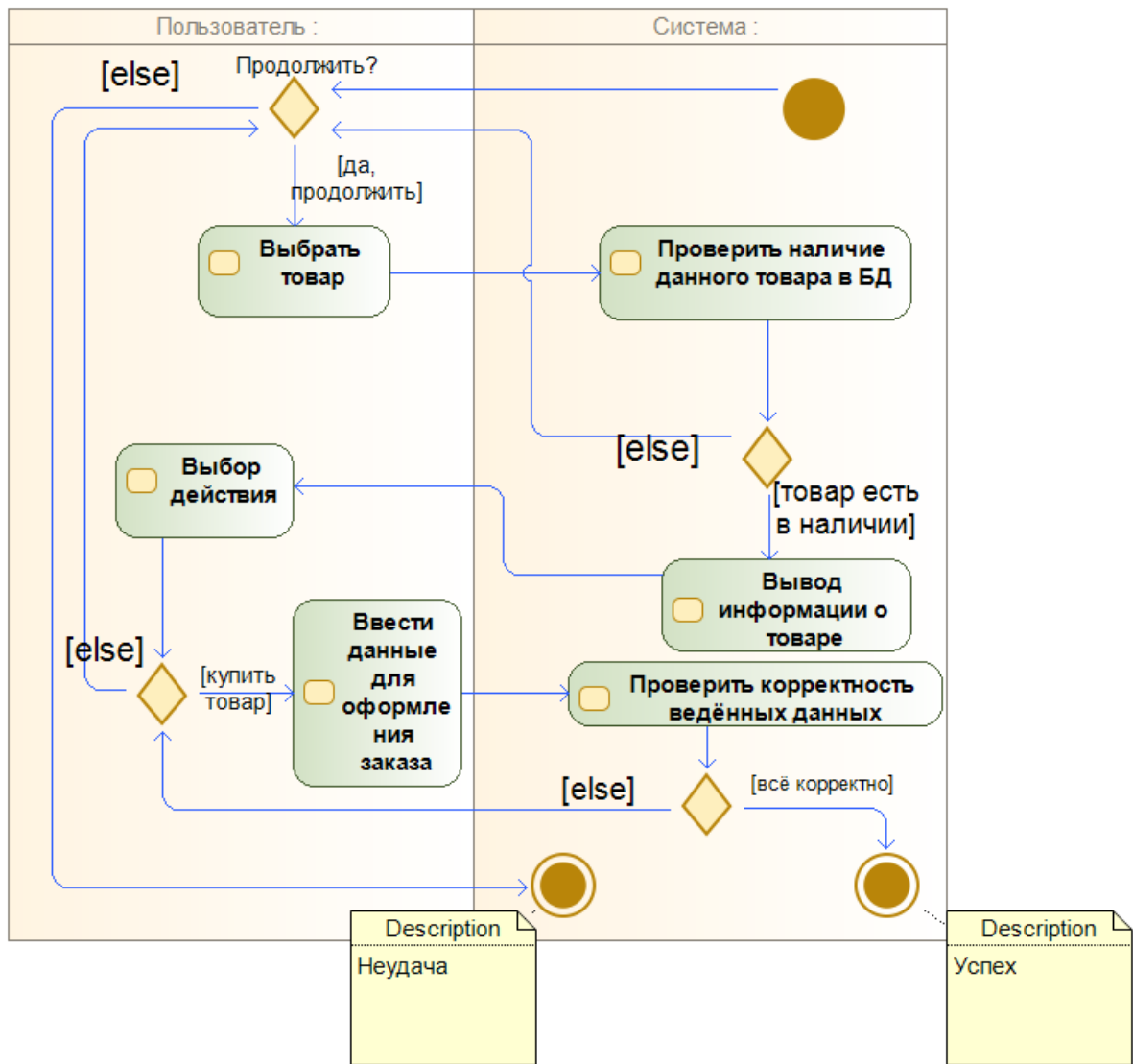
#### 4.8. Дополнительная диаграмма претендентов «Checking Order out»



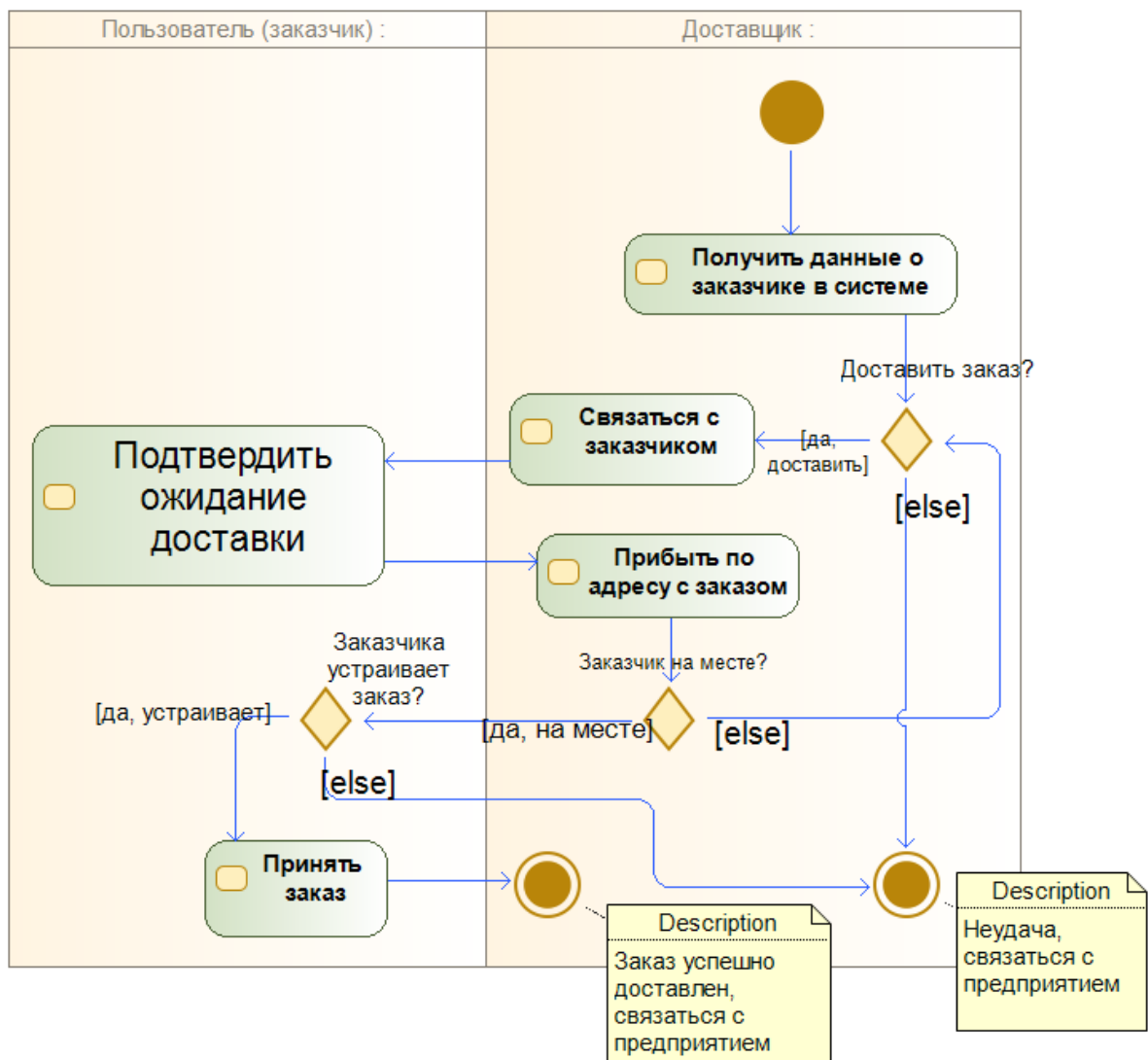
## 4.9. Диаграммы деятельности



Для действия «Войти в систему»



Для действия «Сделать заказ»



Для действия «Доставить заказ»

#### 4.10. Диаграммы состояний



Для состояний системы во время оформления заказа пользователем



Для состояний системы во время создания аккаунта пользователем

## 4.11. Диаграммы классов

Общая диаграмма ключевых абстракций:

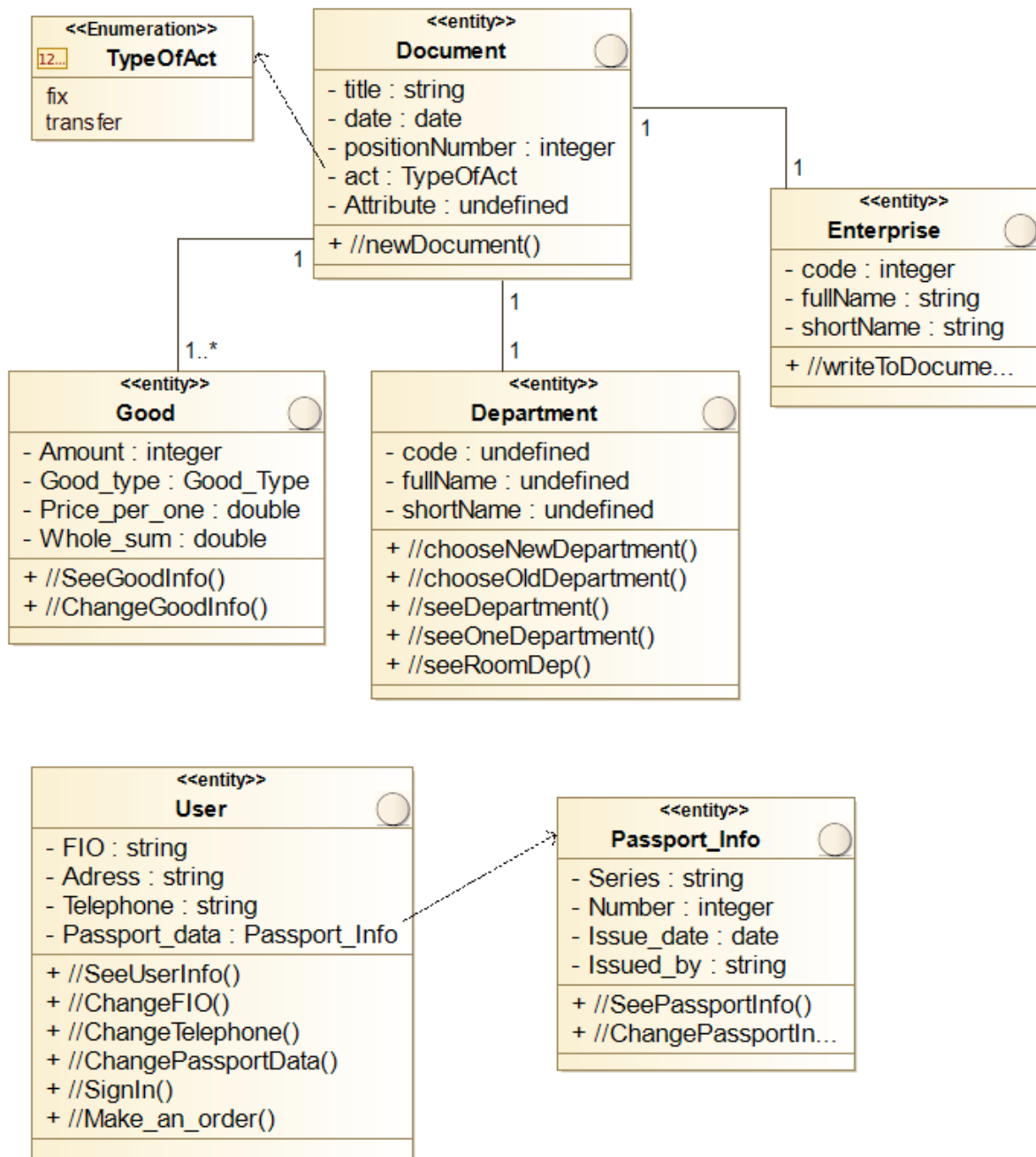


Диаграмма классов для варианта использования «Sign in»:



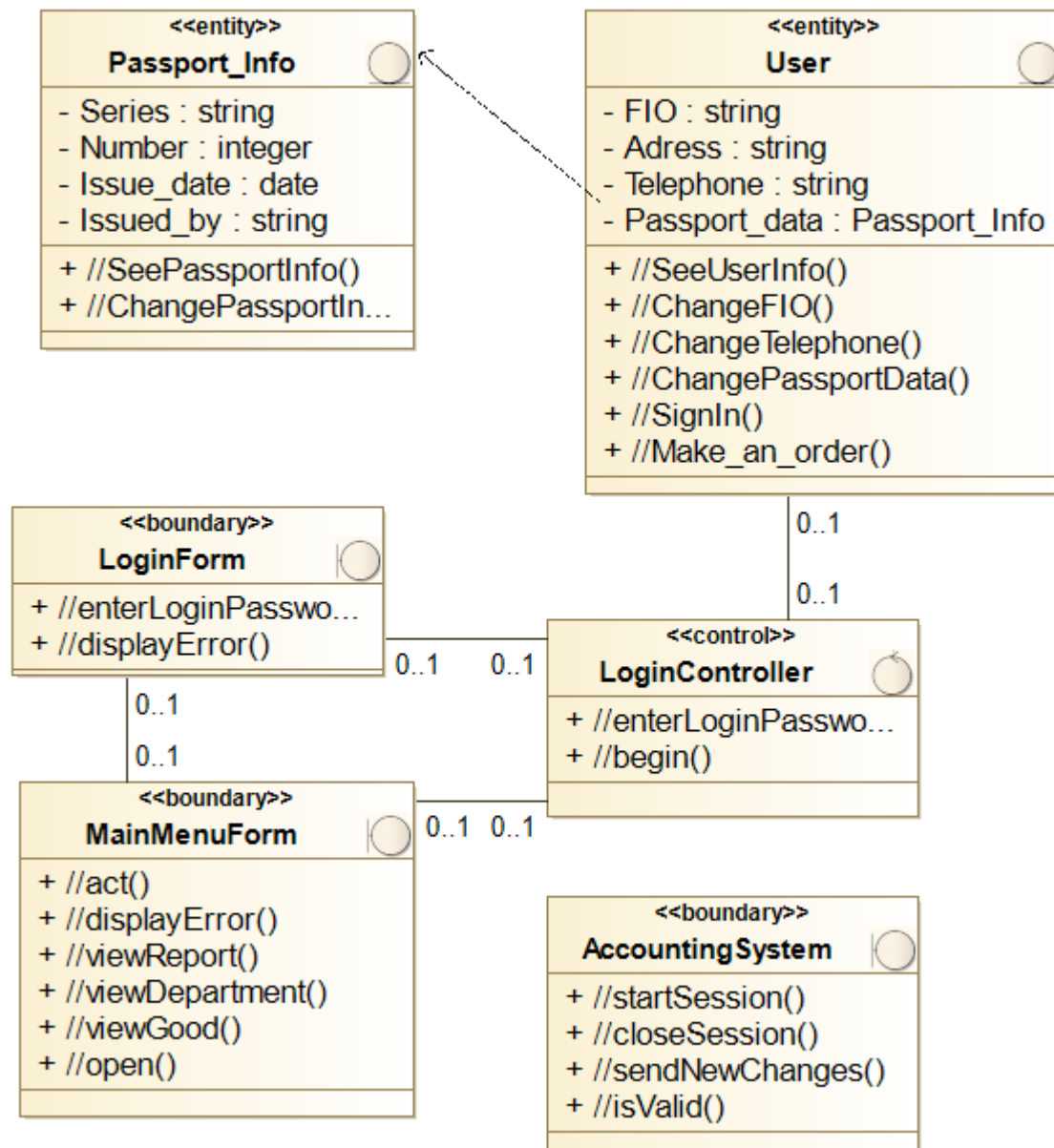


Диаграмма классов для варианта использования «Make an order»:

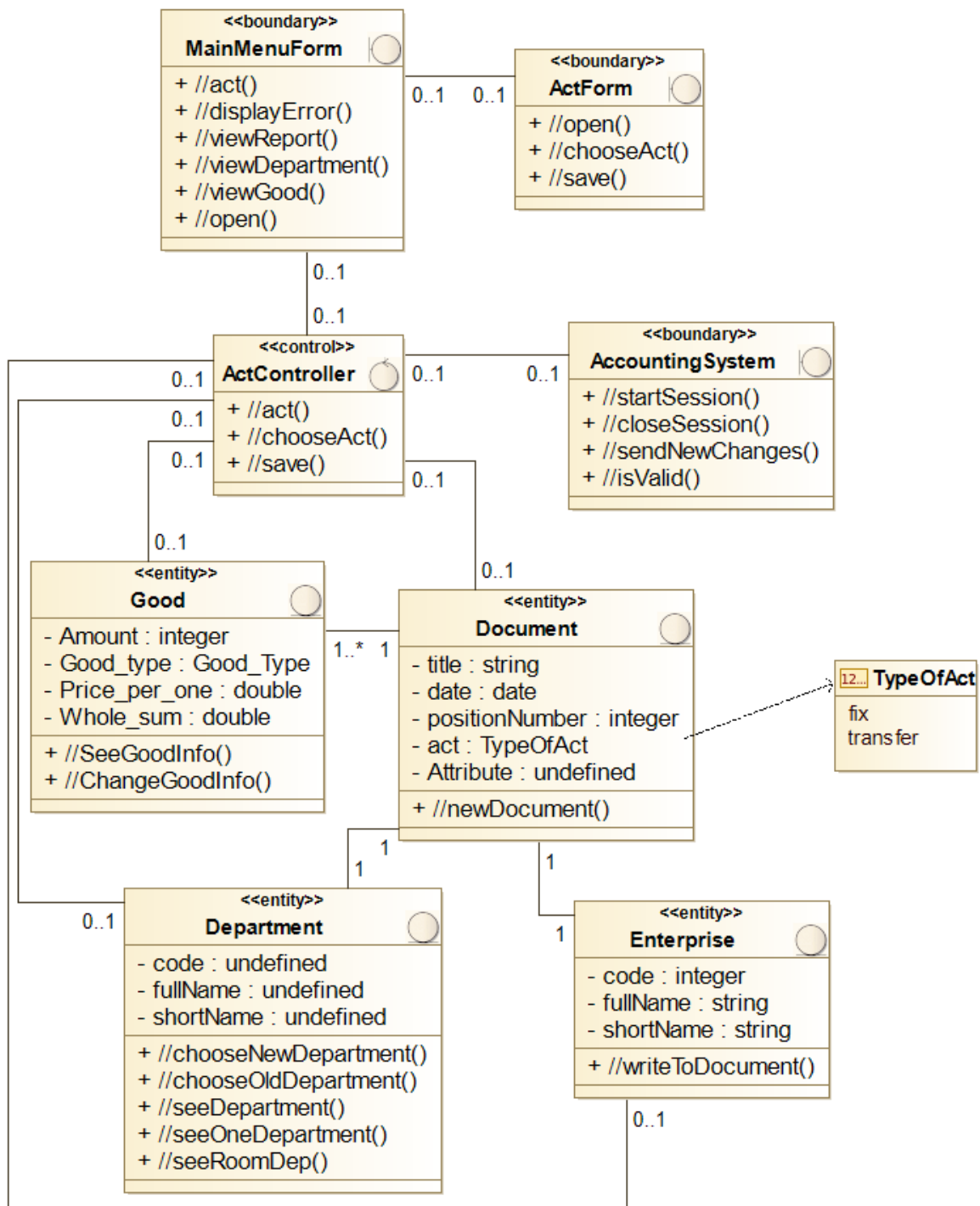


Диаграмма классов для варианта использования «CRUD data about goods»:

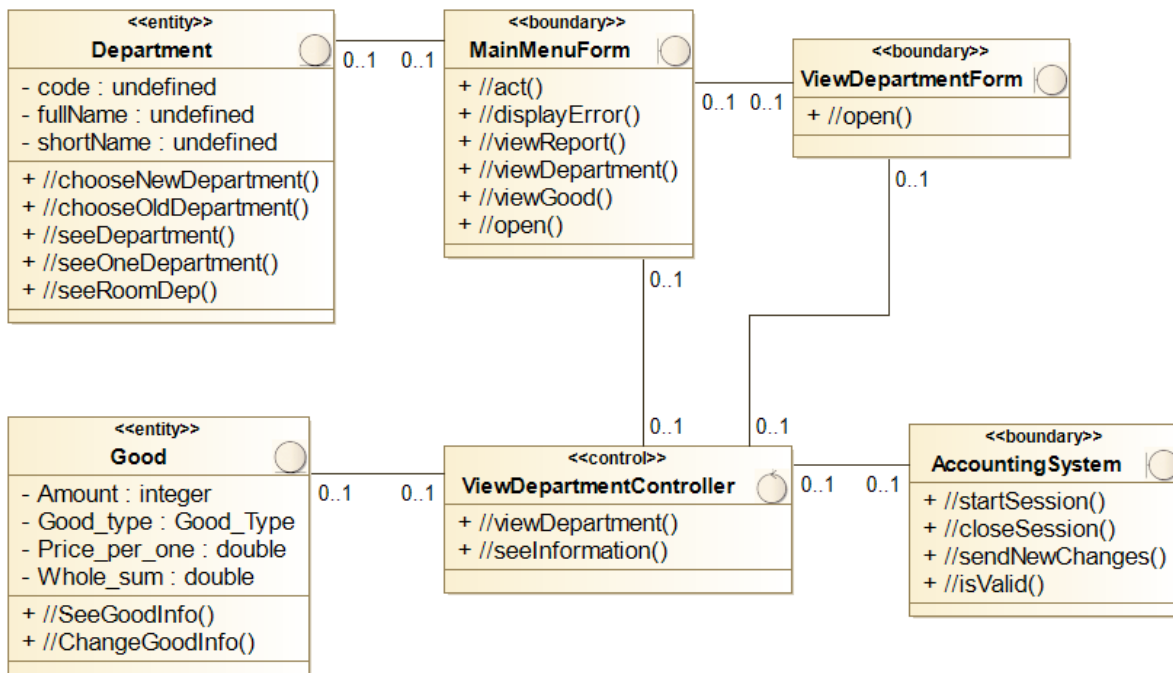


Диаграмма классов для варианта использования «Checkout an order»:

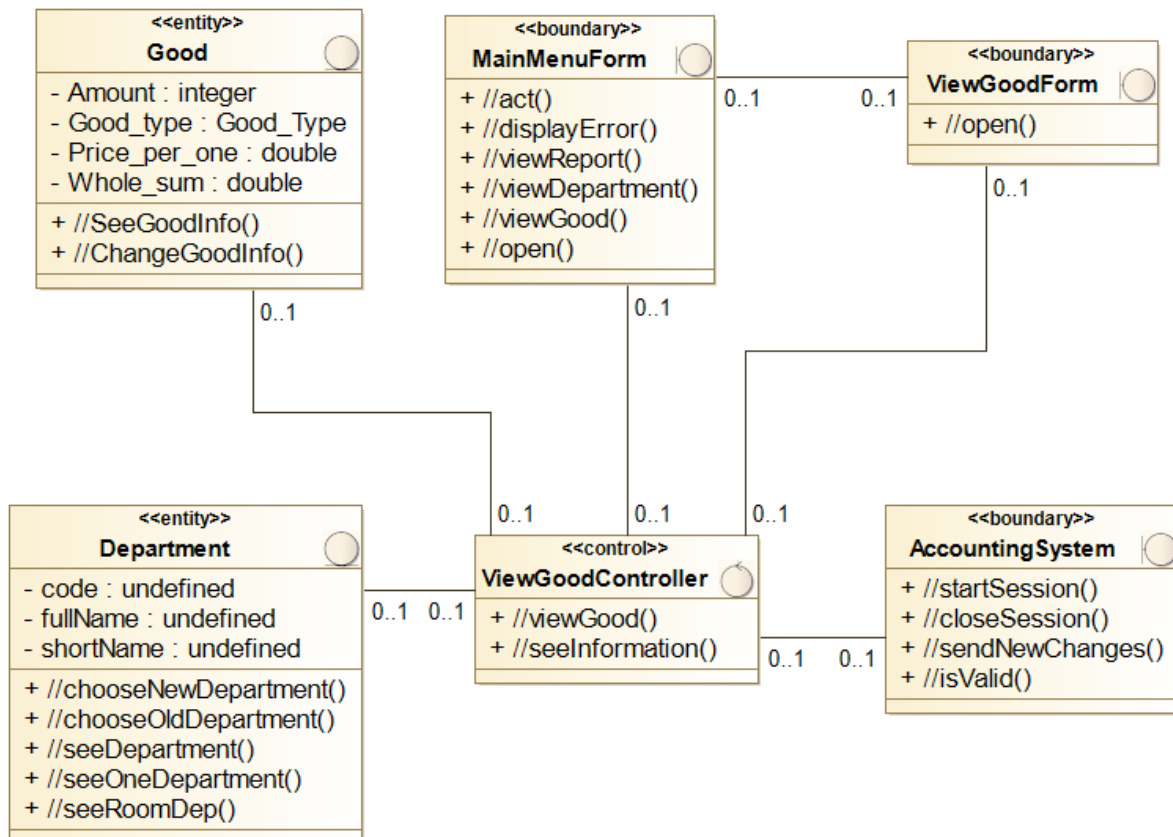
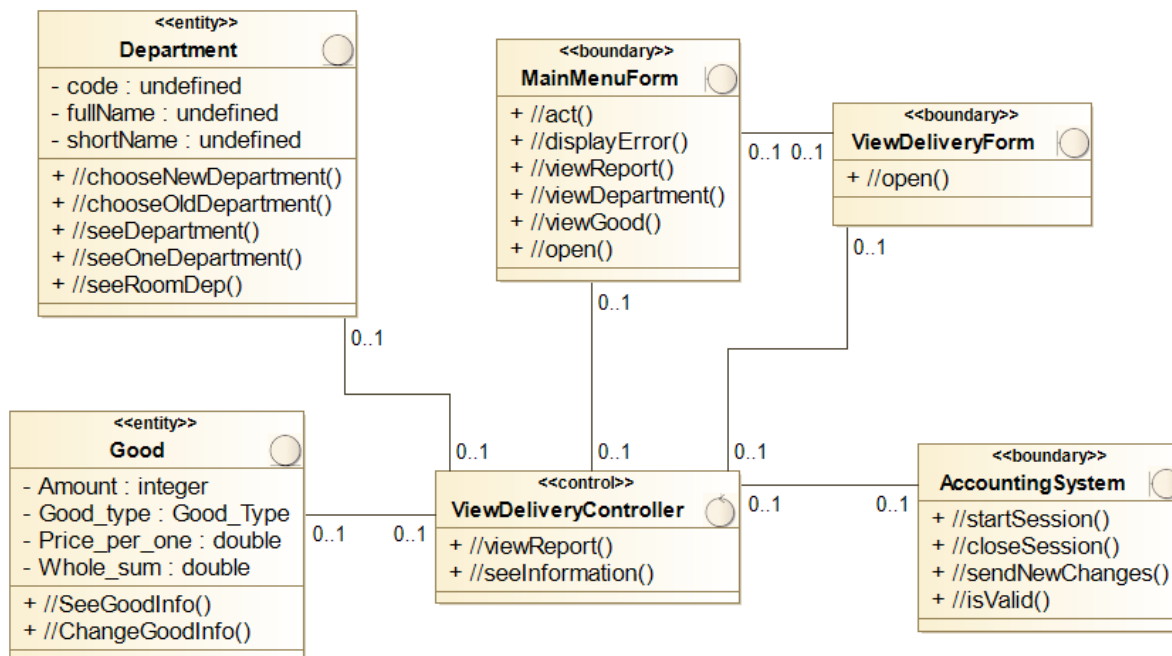


Диаграмма классов для варианта использования «Deliver an order»:



## 4.12. Диаграммы последовательностей

Диаграмма последовательностей для варианта использования «Sign in»:

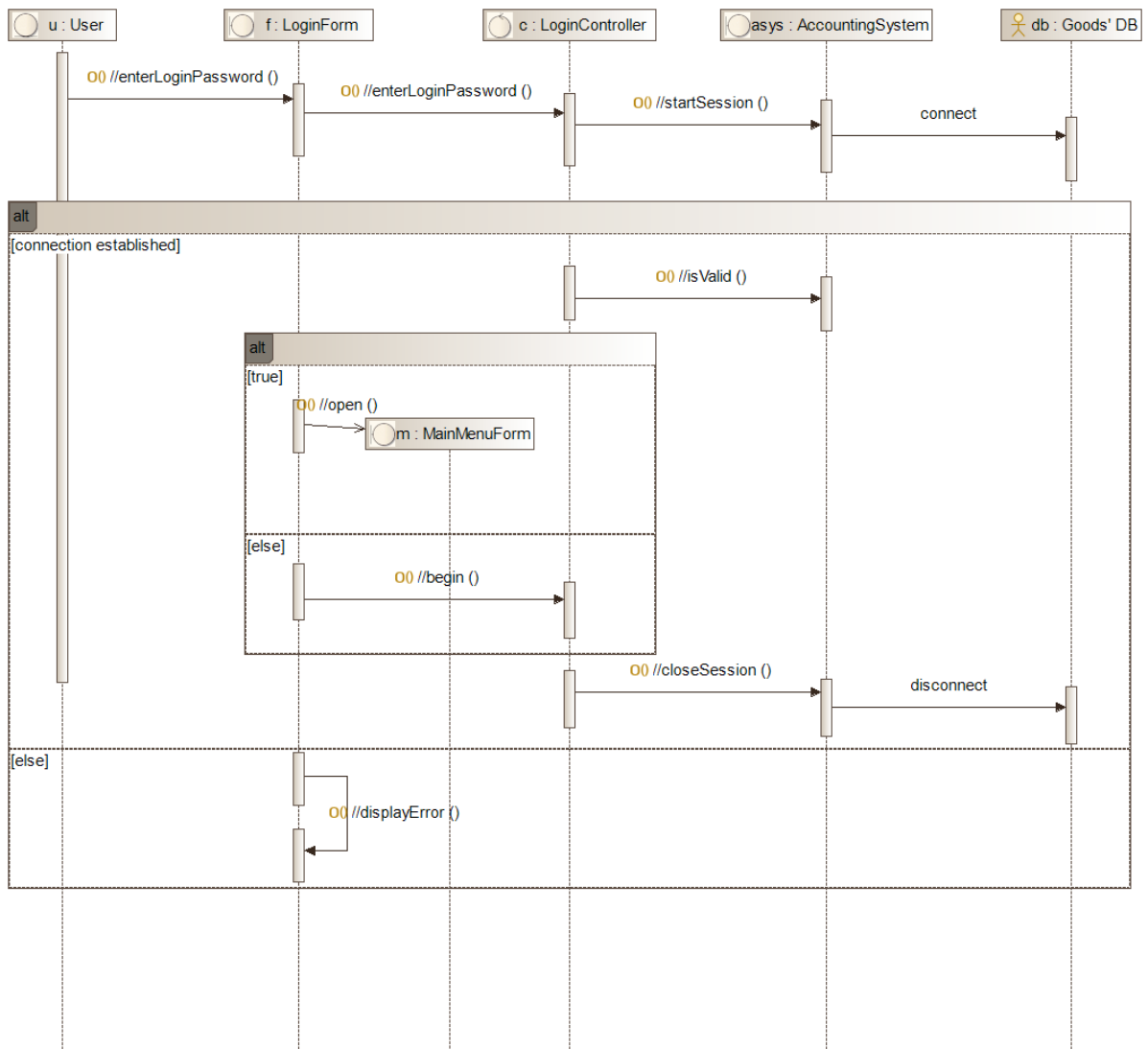


Диаграмма последовательностей для варианта использования «Make an order»:

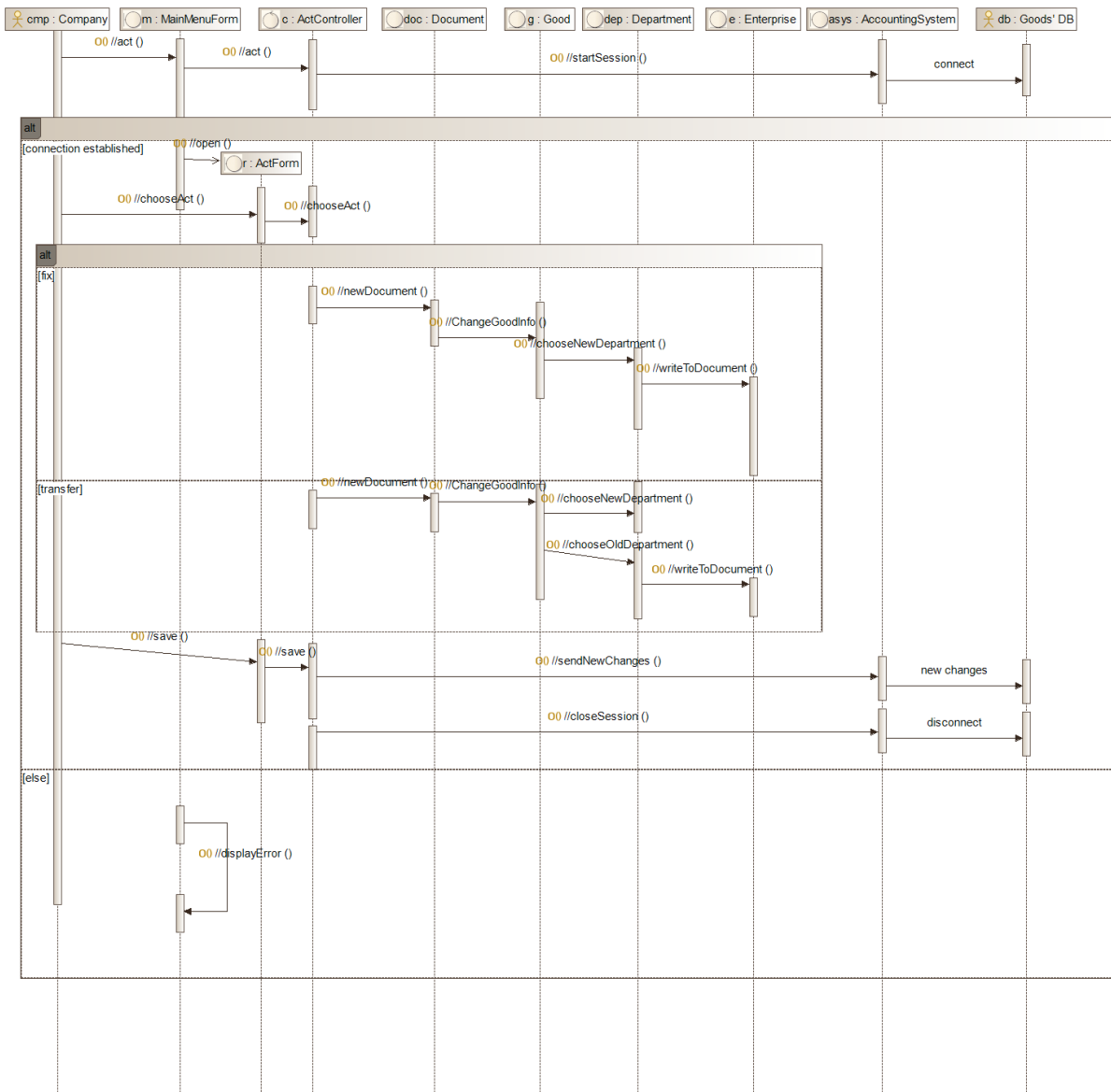


Диаграмма последовательностей для варианта использования «CRUD data about goods»:

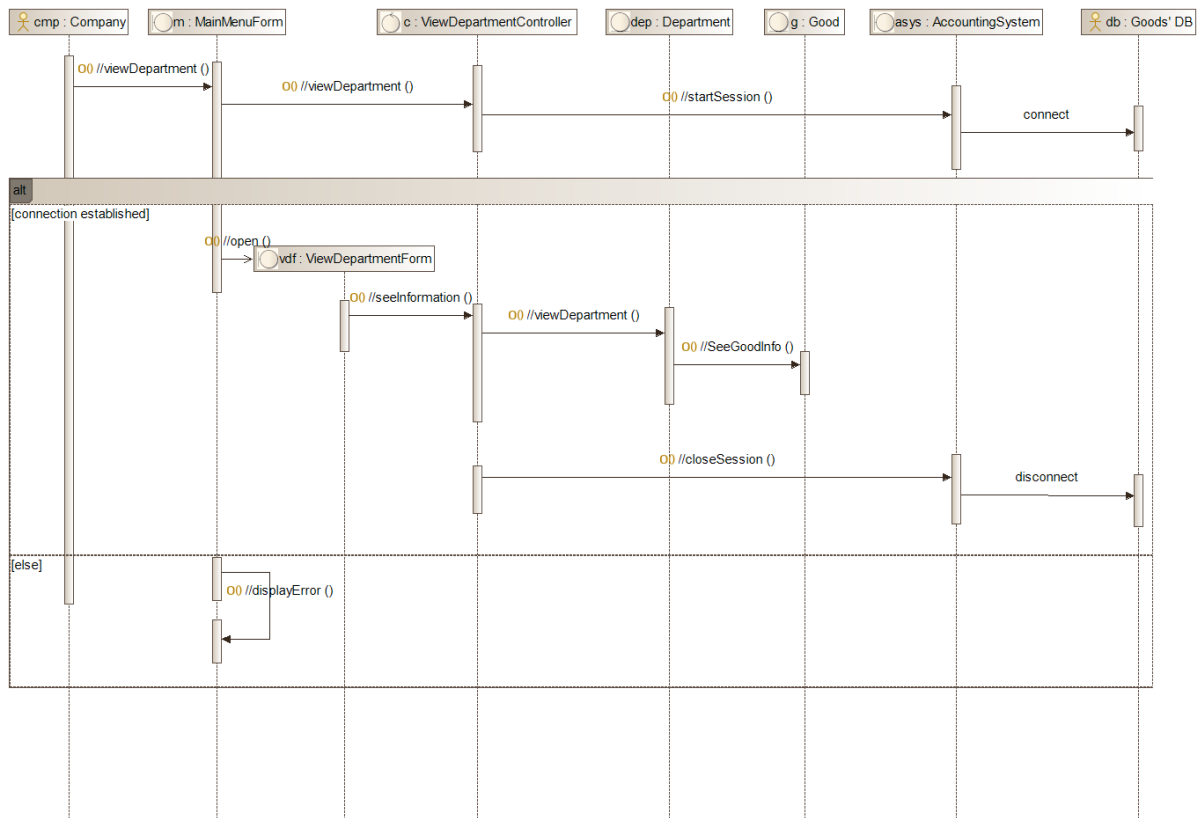


Диаграмма последовательностей для варианта использования «Check out an order»:

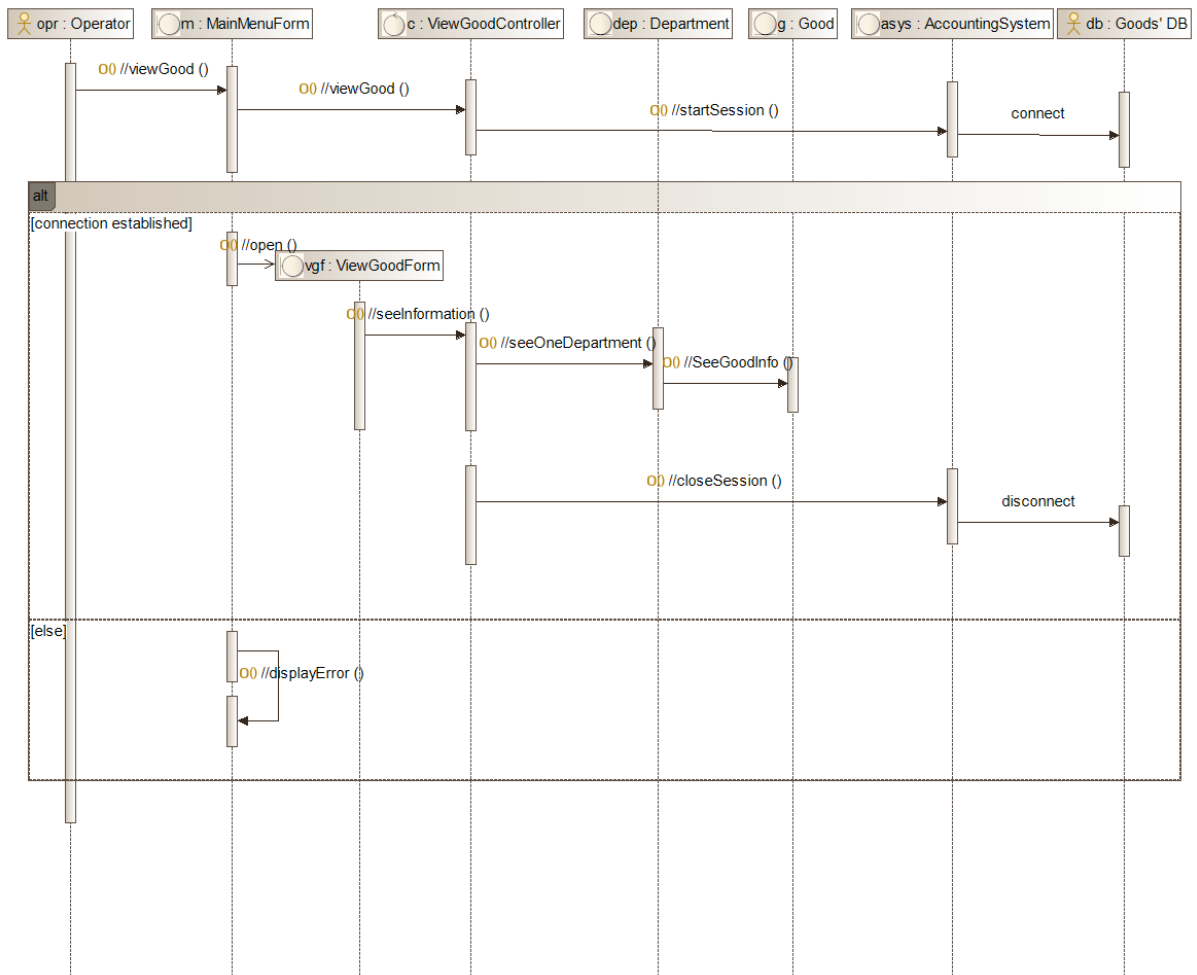
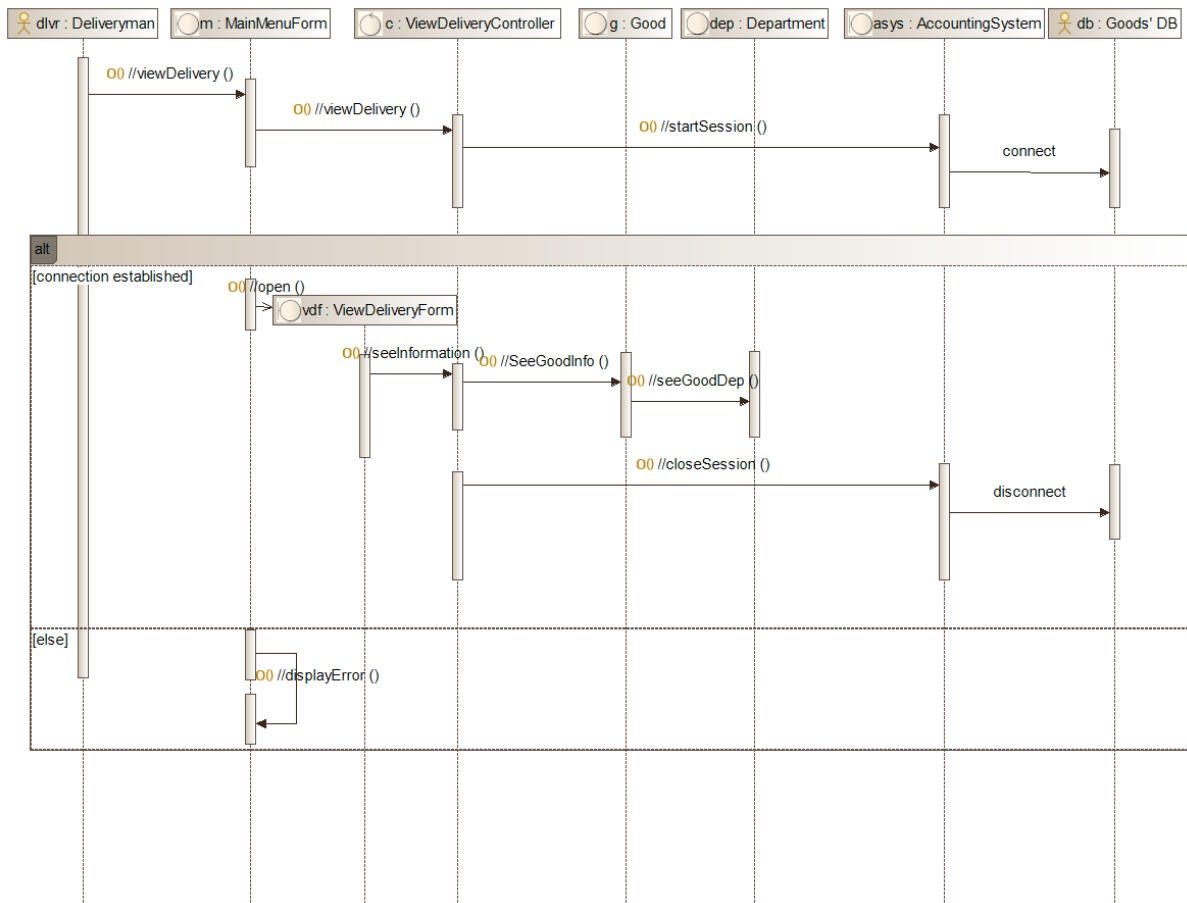


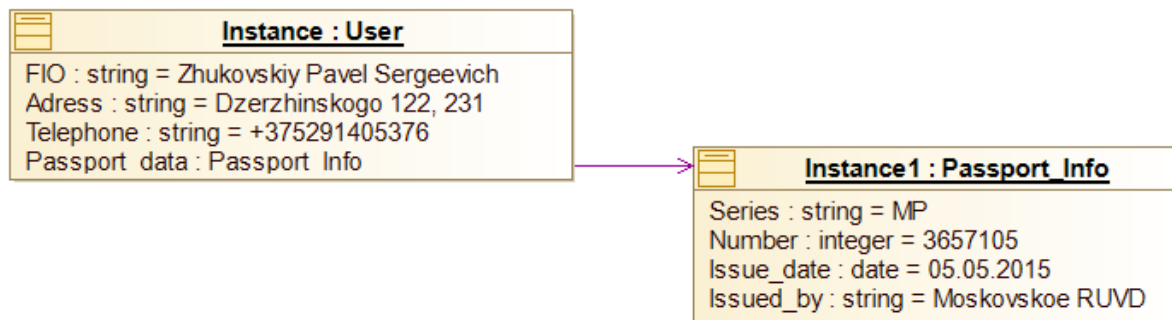
Диаграмма последовательностей для варианта использования «Deliver an order»:





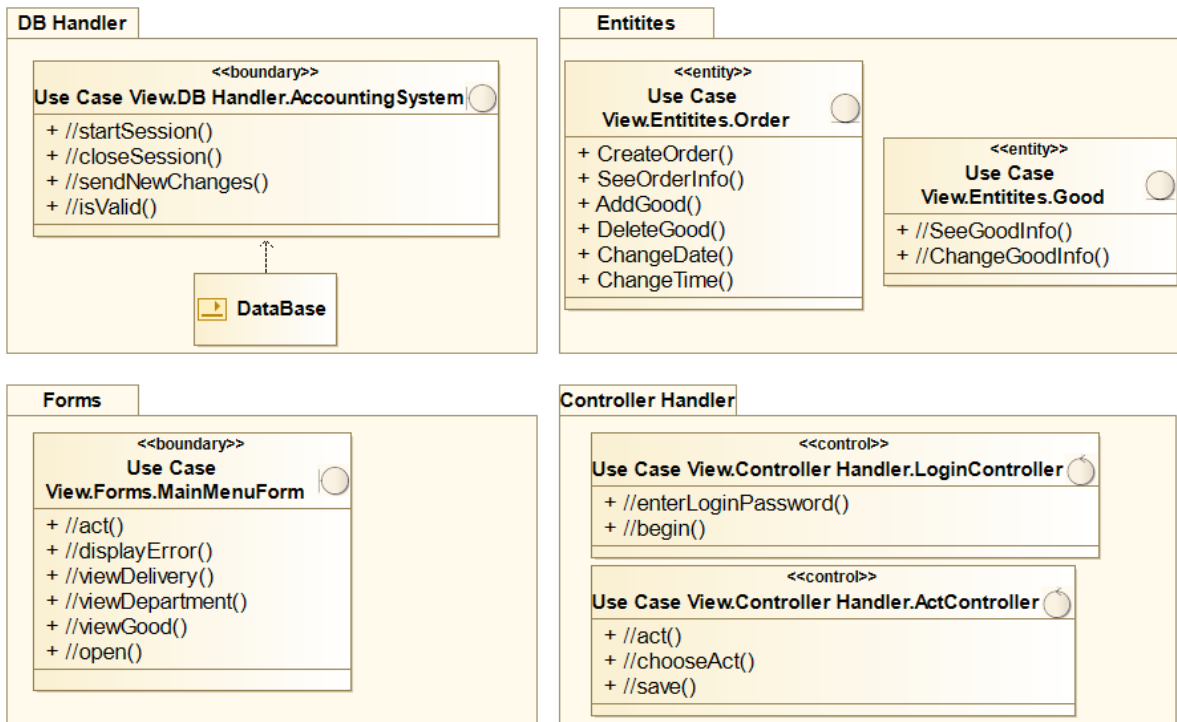
### 4.13. Диаграмма объектов

Диаграмма объектов (на примере классов User и Passport\_info):

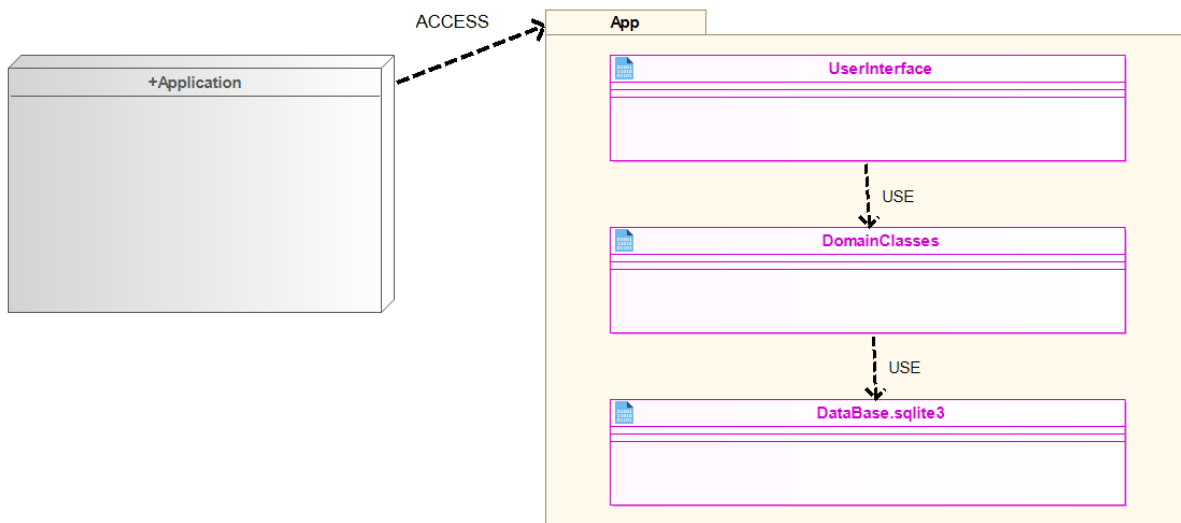


## 5. Проектирование архитектуры и элементов системы

### 5.1. Диаграмма пакетов



### 5.2. Диаграмма развёртывания



### 5.3. Первоначальные представления макета интерфейса

**SIGN IN**

LOGIN

PASSWORD

[Don't have account yet? Sign up!](#)

APPLY

---

**SIGN UP**

Email

LOGIN

PASSWORD

CONFIRM PASSWORD

APPLY

GOOD's SHOP

ACCOUNT INFORMATION

SIGN OUT

Good	Price	Available or not	Buy or not
Good №1	75\$	true	<input type="checkbox"/>
Good №2	82\$	true	<input type="checkbox"/>
Good №3	27\$	false	<input type="checkbox"/>
Good №4	36\$	true	<input type="checkbox"/>

CONFIRM AND GO TO THE ORDER PREPAIRING

ACCOUNT INFORMATION

GO SHOPPING

SIGN OUT

EMAIL

LOGIN

PASSWORD

email

login

password

Change email?

Change login?

Change password?

new email

new login

new password

CONFIRM

CONFIRM

CONFIRM

PREPAIRING THE ORDER

GO SHOPPING

SIGN OUT

YOUR ORDER:

Order information

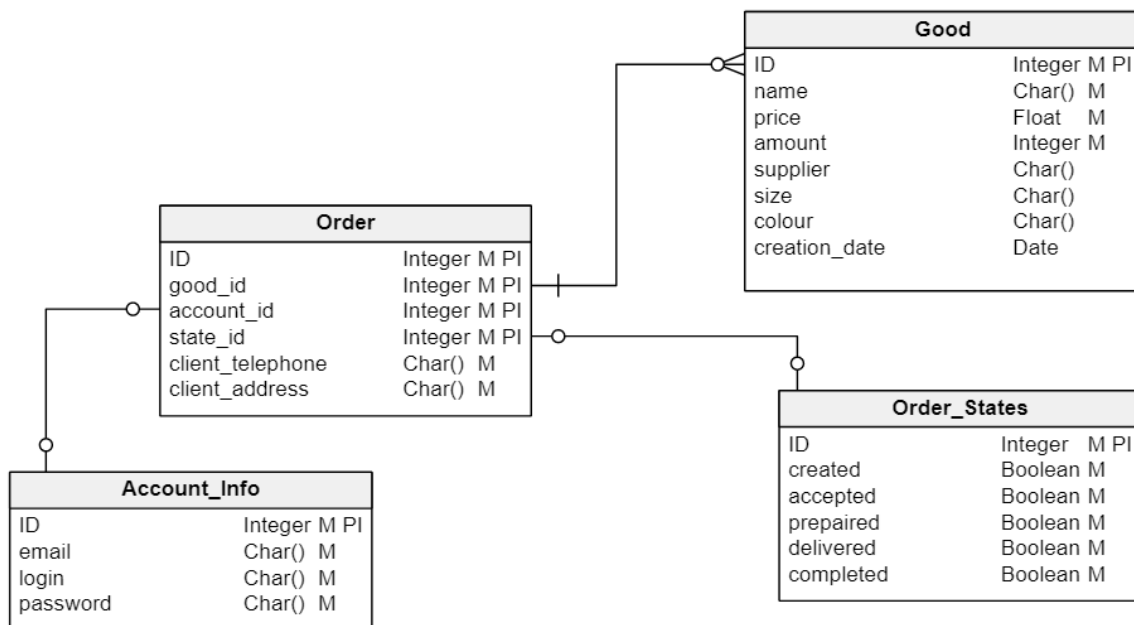
Indicate your contacts and we will accept your order:

Mobile phone:

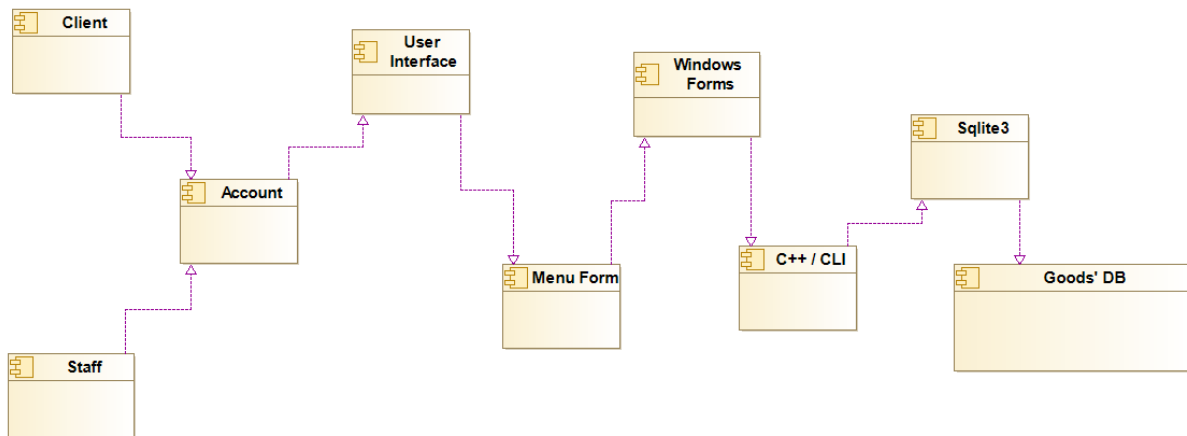
Address:

CONFIRM

#### 5.4. Модель базы данных



## 5.5. Диаграмма компонентов



## 6. Разработка приложения

### 6.1. Сгенерированные классы Modelio

Класс Department:

```
import com.modeliosoft.modelio.javadesigner.annotations.objid;
```

```
@objid ("6d71fc0e-5769-414a-b46e-c329eae2650")
public class Department {
    @objid ("0001d8ee-8c39-4ab7-ba6c-873d5e251f8f")
    private MainMenuForm ;

    @objid ("0bfcd8a0-5ff1-4823-bcc2-3af4458b1160")
    private ViewGoodController ;

    @objid ("596da94f-3486-43c5-ba30-32677095fa70")
    private ViewDeliveryController ;

    @objid ("4132ac34-8b33-4f99-aea7-8a7f56b6c630")
    public void //chooseNewDepartment() {
    }

    @objid ("fd8b2d35-599f-4adb-bd01-6ab70e825c74")
    public void //chooseOldDepartment() {
    }

    @objid ("4e4f0eac-11b4-4d90-9008-987449cda220")
    public void //seeDepartment() {
    }

    @objid ("14158fc0-8211-422f-b0b5-987b5cff9552")
    public void //seeOneDepartment() {
    }

    @objid ("0231cbfc-8f4f-4cab-a9db-f77dfa0b2060")
    public void //seeGoodDep() {
```

```
}  
  
}
```

Класс Document:

```
import java.util.Date;  
import TypeOfAct;  
import com.modeliosoft.modelio.javadesigner.annotations.objid;
```

```
@objid ("f48d87ec-9af5-4fd5-8bd6-e273d36a9de5")  
public class Document {  
    @objid ("b3d9317f-eb28-49e7-94ab-56296f6c1c2f")  
    private String title;  
  
    @objid ("6487168b-8ed9-4ce4-acf9-13a02f0c7ad6")  
    private Date date;  
  
    @objid ("9c166b49-78f0-40fe-9e71-293f141a2e7c")  
    private int positionNumber;  
  
    @objid ("c6b2628c-2faa-41b3-a177-9c0022feafaa")  
    private TypeOfAct act;  
  
    @objid ("7cd1a052-eb90-44fe-bc65-c44c15803587")  
    private Department ;  
  
    @objid ("ba7dd790-4768-4801-94ed-18858478dbd9")  
    private ActController ;  
  
    @objid ("8f9c301b-65fd-4c31-804c-7b3caefe6e10")  
    public void //newDocument() {  
    }  
  
}
```



Класс Enterprise:

```
import com.modeliosoft.modelio.javadesigner.annotations.objid;
```

```
@objid ("a491527a-4cce-4987-a765-41314d9f5273")
public class Enterprise {
    @objid ("9e80292c-2eee-4cde-a83b-8363660ae218")
    private int code;

    @objid ("697bdd20-d941-496e-be19-7b3c1f277cdc")
    private String fullName;

    @objid ("97f3a91c-1419-45e2-9494-f552c9410b86")
    private String shortName;

    @objid ("12b55976-f17c-44f1-9f37-a5af32d4c2f1")
    private Document ;

    @objid ("424544dc-b821-4705-926c-e6d9ad8b71d1")
    public void //writeToDocument() {
    }

}
```

Класс Good:

```
import com.modeliosoft.modelio.javadesigner.annotations.objid;
```

```
@objid ("f50807b8-7a12-4aed-a9f1-e346c8e06003")
public class Good {
    @objid ("29e7e187-1b2f-4aa4-99a0-43fdb01878ca")
    private int Amount;

    @objid ("ca7c3384-5a5f-4f63-910b-ddaecc7241bb")
    private Good_Type Good_type;

    @objid ("7dbe2cdd-29b8-408e-bcf9-19013df5baf3")
```

```

private double Price_per_one;

@objid ("f55a09c2-5018-4e1f-bf5a-d64676dac3dd")
private double Whole_sum;

@objid ("8c8028eb-3c55-431a-b71e-b449c1fd0949")
private Document ;

@objid ("a07dbfb8-5eb8-47b1-8a57-c003e6adbb2e")
private ActionController ;

@objid ("e9f6716b-03a3-4849-9995-9c3279f3cee5")
private ViewDepartmentController ;

@objid ("b7f543f7-8dca-4612-9c5d-7c28b405a5fd")
private ViewGoodController ;

@objid ("91dccaf5-3ea7-48a6-ba7c-cbe0133c49e4")
private ViewDeliveryController ;

@objid ("d97759d3-3a8e-4342-a249-eb65bf5eee97")
public void //SeeGoodInfo() {
}

@objid ("2f68f18d-49f2-4474-8eaf-2f2861855f33")
public void //ChangeGoodInfo() {
}

}

```

Класс Passport\_Info:

```

import java.util.Date;
import com.modeliosoft.modelio.javadesigner.annotations.objid;

```

```

@objid ("aae91818-f1db-438f-aae1-7eea48df2472")

```

```

public class Passport_Info {
    @objid ("5add8200-d8a2-471a-a5fd-ef05cf4b4c0c")
    private String Series;

    @objid ("7d27df32-7c5f-40a0-8967-c1d91ae45b52")
    private int Number;

    @objid ("5924b82f-d3f9-4483-8298-e21642876fcc")
    private Date Issue_date;

    @objid ("33acb579-6c20-4ddd-b724-1aeab53c380e")
    private String Issued_by;

    @objid ("3b44bfe7-da71-45c1-9458-94919447e958")
    public void //SeePassportInfo() {
    }

    @objid ("91ac6431-70b6-41c6-9faa-32f4202736ce")
    public void //ChangePassportInfo() {
    }

}

```

Класс TypeOfAct:

```
import com.modeliosoft.modelio.javadesigner.annotations.objid;
```

```

@objid ("94805f57-a140-4b7d-9937-3ad9cd828e62")
public enum TypeOfAct {
    fix,
    transfer;
}

```

Класс User:

```

import java.util.ArrayList;
import java.util.List;

```

```

import com.modeliosoft.modelio.javadesigner.annotations.objid;

@objid ("65b39c4a-fdce-48b4-8b70-9ecfa27c4429")
public class User {
    @objid ("9dd10bbf-46fa-40ee-8d96-60b2d15673c8")
    private String FIO;

    @objid ("1b14e6ee-389f-4415-a626-663afe564a7f")
    private String Adress;

    @objid ("22648efc-45ba-46ee-9e2e-4653dfd75c85")
    private String Telephone;

    @objid ("89989e34-d927-4ecb-b03a-0494f41f3f4b")
    private Passport_Info Passport_data;

    @objid ("4ab1f2c5-3f69-4737-a3cd-2b61b00ace85")
    private List<Order> Make an order = new ArrayList<Order> ();

    @objid ("edf40fe1-b118-4066-88ac-2cf44ce45bb0")
    public void //SeeUserInfo() {
    }

    @objid ("4ba6180c-b1f4-4f2c-abc0-c7d8eb88cdf6")
    public void //ChangeFIO() {
    }

    @objid ("441b57aa-b75e-4907-9797-7fbcd9fec0ed")
    public void //ChangeTelephone() {
    }

    @objid ("5a20e520-c929-4fd0-bbb1-bdea8dfed13b")
    public void //ChangePassportData() {
    }

    @objid ("9d561cd0-8cf6-49c1-9c7c-3817c78dae9a")

```

```

public void //SignIn() {
}

@objid ("57cc0f81-84c8-4411-981c-ed1eec7eac67")
public void //Make_an_order() {
}

}

```

## 6.2. Исходный код приложения

```

from tkinter import *
from tkinter import messagebox
import sqlite3

CurrentID = 1
CurrentName = ""
CurrentPrice = 0.0
CurrentAmount = 0

CurrentGood = [0, "", 0.0, 0]

CurrentOrder = []

Window_SignIn = Tk()
Window_SignIn.withdraw()

Window_SignUp = Toplevel(Window_SignIn)
Window_SignUp.withdraw()

Window_OnlineShop = Toplevel(Window_SignUp)
Window_OnlineShop.withdraw()

Window_AccountInfo = Toplevel(Window_OnlineShop)
Window_AccountInfo.withdraw()

Window_PreparingOrder = Toplevel(Window_AccountInfo)
Window_PreparingOrder.withdraw()

```

```

SignIn_Email = StringVar()
SignIn_Password = StringVar()

SignUp_Email = StringVar()
SignUp_Password = StringVar()
SignUp_PasswordConfirm = StringVar()

AccountInfo_NewEmail = StringVar()
AccountInfo_NewPassword = StringVar()

HowManyWasTaken = StringVar()

MobilePhone = StringVar()

# --- БЛОК РАБОТЫ С БД ---
conn = sqlite3.connect("DataBase.db") # Подключение к БД с именем DataBase.db
cursor = conn.cursor()
# --- БЛОК РАБОТЫ С БД ---

# --- БЛОК РАБОТЫ С GUI ---
def SignIn_ButtonToSignUp_Clicked():
    Window_SignIn.withdraw()
    Window_SignUp_PowerOn()

def SignIn_ButtonConfirm_Clicked():
    sql1 = f"SELECT * FROM ACCOUNTS WHERE email LIKE '{str(SignIn_Email.get())}'"
    cursor.execute(sql1)
    if cursor.fetchall():
        sql2 = f"SELECT password FROM ACCOUNTS WHERE email LIKE '{str(SignIn_Email.get())}'"
        cursor.execute(sql2)
        if str(SignIn_Password.get()) == cursor.fetchall()[0][0]:
            Window_SignIn.withdraw()
            Window_OnlineShop_PowerOn()
        else:
            messagebox.showinfo("Online Shop - Sign In", f"Incorrect password for this account!")
    else:
        messagebox.showinfo("Online Shop - Sign In", f"No email like

```

```

{str(SignIn_Email.get())} in DB!")

def Window_SignIn_PowerOn():
    Window_SignIn.deiconify()
    Window_SignIn.geometry('1500x800')
    Window_SignIn.resizable(width=False, height=False)
    Window_SignIn["bg"] = "#98FB98"
    Window_SignIn.title("Online Shop - Sign In")

    SignIn_Lbl1 = Label(Window_SignIn, text="SIGN IN", font=("Arial Bold", 50),
bg="#98FB98")
    SignIn_Lbl1.place(relx=0.5, rely=0.15, anchor="c")

    SignIn_Lbl2 = Label(Window_SignIn, text="Enter Email:", font=("Arial Bold",
36), bg="#98FB98")
    SignIn_Lbl2.place(relx=0.2, rely=0.3, anchor="c")

    SignIn_Lbl3 = Label(Window_SignIn, text="Enter Password:", font=("Arial Bold",
36), bg="#98FB98")
    SignIn_Lbl3.place(relx=0.17, rely=0.4, anchor="c")

    SignIn_EmailTxt = Entry(Window_SignIn, width=20, bd=5, font=("Arial Bold",
36), textvariable=SignIn_Email)
    SignIn_EmailTxt.place(relx=0.5, rely=0.3, anchor="c")

    SignIn_PasswordTxt = Entry(Window_SignIn, width=20, bd=5, font=("Arial
Bold", 36), show='*',
textvariable=SignIn_Password)
    SignIn_PasswordTxt.place(relx=0.5, rely=0.4, anchor="c")

    SignIn_ButtonToSignUp = Button(Window_SignIn, text="Don't have account yet?
Sign Up!", font=("Arial Bold", 8), bd=5,
background="#3CB371",
command=SignIn_ButtonToSignUp_Clicked)
    SignIn_ButtonToSignUp.place(relx=0.5, rely=0.5, anchor="c")

    SignIn_ButtonConfirm = Button(Window_SignIn, text="CONFIRM", font=("Arial
Bold", 24), bd=10, background="#3CB371",
command=SignIn_ButtonConfirm_Clicked)
    SignIn_ButtonConfirm.place(relx=0.5, rely=0.7, anchor="c")

```

```

Window_SignIn.mainloop()

def SignUp_ButtonToSignIn_Clicked():
    Window_SignUp.withdraw()
    Window_SignIn_PowerOn()

def SignUp_ButtonConfirm_Clicked():
    if '@' in str(SignUp_Email.get()):
        if 7 < len((str(SignUp_Password.get()))) < 17:
            if str(SignUp_Password.get()) == str(SignUp_PasswordConfirm.get()):
                cursor.execute(f"INSERT INTO ACCOUNTS VALUES
('{'str(SignUp_Email.get())}', {'str(SignUp_Password.get())'})")
                conn.commit()
                messagebox.showinfo("Online Shop - Sign Up", "Account created
successfully and added to DB!")
                Window_SignUp.withdraw()
                Window_SignIn_PowerOn()
            else:
                messagebox.showinfo("Online Shop - Sign Up", f"Passwords do not
match!")
        else:
            messagebox.showinfo("Online Shop - Sign Up", f"Password must be from 8
to 16 characters!")
        else:
            messagebox.showinfo("Online Shop - Sign Up", f"Entered email
{'str(SignUp_Email.get())}' is not valid!")

def Window_SignUp_PowerOn():
    Window_SignUp.deiconify()
    Window_SignUp.geometry('1500x800')
    Window_SignUp.resizable(width=False, height=False)
    Window_SignUp["bg"] = "#DDA0DD"
    Window_SignUp.title("Online Shop - Sign Up")

    SignUp_Lbl1 = Label(Window_SignUp, text="SIGN UP", font=("Arial Bold", 50),
bg="#DDA0DD")
    SignUp_Lbl1.place(relx=0.5, rely=0.1, anchor="c")

```



```

SignUp_Lbl2 = Label(Window_SignUp, text="Enter Email:", font=("Arial Bold",
36), bg="#DDA0DD")
SignUp_Lbl2.place(relx=0.2, rely=0.25, anchor="c")

SignUp_Lbl3 = Label(Window_SignUp, text="Enter Password:", font=("Arial
Bold", 36), bg="#DDA0DD")
SignUp_Lbl3.place(relx=0.17, rely=0.35, anchor="c")

SignUp_Lbl4 = Label(Window_SignUp, text="Confirm Password:", font=("Arial
Bold", 36), bg="#DDA0DD")
SignUp_Lbl4.place(relx=0.15, rely=0.45, anchor="c")

SignUp_EmailTxt = Entry(Window_SignUp, width=20, bd=5, font=("Arial Bold",
36), textvariable=SignUp_Email)
SignUp_EmailTxt.place(relx=0.5, rely=0.25, anchor="c")

SignUp_PasswordTxt = Entry(Window_SignUp, width=20, bd=5, font=("Arial
Bold", 36), show='*',
                        textvariable=SignUp_Password)
SignUp_PasswordTxt.place(relx=0.5, rely=0.35, anchor="c")

SignUp_PasswordConfirmTxt = Entry(Window_SignUp, width=20, bd=5,
font=("Arial Bold", 36), show='*',
                        textvariable=SignUp_PasswordConfirm)
SignUp_PasswordConfirmTxt.place(relx=0.5, rely=0.45, anchor="c")

SignUp_ButtonToSignIn = Button(Window_SignUp, text="Already have account?
Sign In!", font=("Arial Bold", 8), bd=5,
                        background="#DA70D6",
command=SignUp_ButtonToSignIn_Clicked)
SignUp_ButtonToSignIn.place(relx=0.5, rely=0.55, anchor="c")

SignUp_ButtonConfirm = Button(Window_SignUp, text="CONFIRM",
font=("Arial Bold", 24), bd=10, background="#DA70D6",
                        command=SignUp_ButtonConfirm_Clicked)
SignUp_ButtonConfirm.place(relx=0.5, rely=0.7, anchor="c")

Window_SignUp.mainloop()

def OnlineShop_ButtonSignOut_Clicked():

```

```

Window_OnlineShop.withdraw()
Window_SignIn_PowerOn()

def OnlineShop_ButtonAccountInfo_Clicked():
    Window_OnlineShop.withdraw()
    Window_AccountInfo_PowerOn()

def OnlineShop_ButtonConfirmOrder_Clicked():
    Window_OnlineShop.withdraw()
    Window_PreParingOrder_PowerOn()

def OnlineShop_ButtonPreviousGood_Clicked():
    Window_OnlineShop.withdraw()
    global CurrentID
    CurrentID -= 1
    Window_OnlineShop_PowerOn()

def OnlineShop_ButtonNextGood_Clicked():
    Window_OnlineShop.withdraw()
    global CurrentID
    CurrentID += 1
    Window_OnlineShop_PowerOn()

def OnlineShop_ButtonTakeToBasket_Clicked():
    global CurrentID
    global CurrentName
    global CurrentPrice
    global CurrentGood
    global CurrentOrder
    CurrentGood[0] = CurrentID
    CurrentGood[1] = CurrentName
    CurrentGood[2] = CurrentPrice
    CurrentGood[3] = int(HowManyWasTaken.get())
    CurrentOrder.append(CurrentGood.copy())
    Window_OnlineShop.withdraw()
    Window_OnlineShop_PowerOn()

def Window_OnlineShop_PowerOn():
    Window_OnlineShop.deiconify()
    Window_OnlineShop.geometry('1500x800')
    Window_OnlineShop.resizable(width=False, height=False)

```

```

Window_OnlineShop["bg"] = "#F0E68C"
Window_OnlineShop.title("Online Shop - Catalog")

cursor.execute("""SELECT COUNT(*) as count FROM GOODS""")
GoodsWholeAmount = cursor.fetchall()[0][0]

PreviousGoodButtonState = "normal"
NextGoodButtonState = "normal"

if CurrentID == 1:
    PreviousGoodButtonState = "disabled"
if CurrentID == GoodsWholeAmount:
    NextGoodButtonState = "disabled"
cursor.execute(f""""SELECT * FROM GOODS WHERE ID LIKE {CurrentID}""")
GoodInfo = cursor.fetchall()[0]
global CurrentName
CurrentName = GoodInfo[1]
global CurrentPrice
CurrentPrice = GoodInfo[2]
global CurrentAmount
CurrentAmount = GoodInfo[3]

TextForBasket = []
global CurrentOrder
k = 0
for GoodSort in CurrentOrder:
    k += 1
    TextForBasket.append(str(k) + " " + GoodSort[1] + " ( " + "ID = " +
str(GoodSort[0]) + " ), " +
                        str(GoodSort[3]) + " items, whole price - " +
str(GoodSort[2]*GoodSort[3]) + "$\n")

OnlineShop_Lbl1 = Label(Window_OnlineShop, text="GOOD's SHOP",
font=("Arial Bold", 50), bg="#F0E68C")
OnlineShop_Lbl1.place(relx=0.2, rely=0.08, anchor="c")

OnlineShop_ButtonAccountInfo = Button(Window_OnlineShop,
text="ACCOUNT INFORMATION", font=("Arial Bold", 24), bd=10,
background="#DAA520",
command=OnlineShop_ButtonAccountInfo_Clicked)
OnlineShop_ButtonAccountInfo.place(relx=0.65, rely=0.08, anchor="c")

```

```

OnlineShop_ButtonSignOut = Button(Window_OnlineShop, text="SIGN OUT",
font=("Arial Bold", 24), bd=10,
background="#DAA520",
command=OnlineShop_ButtonSignOut_Clicked)
OnlineShop_ButtonSignOut.place(relx=0.9, rely=0.08, anchor="c")

OnlineShop_Lb_ID = Label(Window_OnlineShop, text="Good's ID", font=("Arial
Bold", 24), bg="#FFA500", padx=50,
pady=20, relief="solid")
OnlineShop_Lb_ID.place(relx=0.1, rely=0.35, anchor="c")

OnlineShop_Lb_ID_Value = Label(Window_OnlineShop, text=str(CurrentID),
font=("Arial Bold", 12), bg="#F0E68C",
padx=50, pady=20)
OnlineShop_Lb_ID_Value.place(relx=0.1, rely=0.443, anchor="c")

OnlineShop_Lb_Name = Label(Window_OnlineShop, text="Good's Name",
font=("Arial Bold", 24), bg="#FFA500", padx=50,
pady=20, relief="solid")
OnlineShop_Lb_Name.place(relx=0.28, rely=0.35, anchor="c")

OnlineShop_Lb_Name_Value = Label(Window_OnlineShop,
text=str(CurrentName), font=("Arial Bold", 12), bg="#F0E68C",
padx=50, pady=20)
OnlineShop_Lb_Name_Value.place(relx=0.28, rely=0.443, anchor="c")

OnlineShop_Lb_Price = Label(Window_OnlineShop, text="Price", font=("Arial
Bold", 24), bg="#FFA500", padx=50,
pady=20, relief="solid")
OnlineShop_Lb_Price.place(relx=0.438, rely=0.35, anchor="c")

OnlineShop_Lb_Price_Value = Label(Window_OnlineShop,
text=str(CurrentPrice)+"$", font=("Arial Bold", 12),
bg="#F0E68C", padx=50, pady=20)
OnlineShop_Lb_Price_Value.place(relx=0.44, rely=0.443, anchor="c")

OnlineShop_Lb_Amount = Label(Window_OnlineShop, text="Amount in stock",
font=("Arial Bold", 24), bg="#FFA500",
padx=50, pady=20, relief="solid")
OnlineShop_Lb_Amount.place(relx=0.607, rely=0.35, anchor="c")

```

```

OnlineShop_Lb_Amount_Value = Label(Window_OnlineShop,
text=str(CurrentAmount), font=("Arial Bold", 12),
                                bg="#F0E68C", padx=50, pady=20)
OnlineShop_Lb_Amount_Value.place(relx=0.61, rely=0.443, anchor="c")

OnlineShop_Lb_Take = Label(Window_OnlineShop, text="Take some?",
font=("Arial Bold", 24), bg="#00BFFF",
                                padx=50, pady=20, relief="solid")
OnlineShop_Lb_Take.place(relx=0.85, rely=0.35, anchor="c")

OnlineShop_SpinBox = Spinbox(Window_OnlineShop, from_=1,
to=CurrentAmount, width=5, bg="#00FFFF", bd=10,
                                textvariable=HowManyWasTaken)
OnlineShop_SpinBox.place(relx=0.78, rely=0.45, anchor="c")

OnlineShop_ButtonTakeToBasket = Button(Window_OnlineShop, text="TAKE
TO BASKET", font=("Arial Bold", 12), bd=10,
                                background="#00FFFF",
command=OnlineShop_ButtonTakeToBasket_Clicked)
OnlineShop_ButtonTakeToBasket.place(relx=0.88, rely=0.45, anchor="c")

OnlineShop_ButtonConfirmOrder = Button(Window_OnlineShop,
text="CONFIRM AND GO TO THE ORDER PREPARING",
                                font=("Arial Bold", 24), bd=10, background="#DAA520",
                                command=OnlineShop_ButtonConfirmOrder_Clicked)
OnlineShop_ButtonConfirmOrder.place(relx=0.3, rely=0.9, anchor="c")

OnlineShop_Lbl2 = Label(Window_OnlineShop, text="PREVIOUS GOOD",
font=("Arial Bold", 36), bg="#F0E68C")
OnlineShop_Lbl2.place(relx=0.15, rely=0.58, anchor="c")

OnlineShop_Lbl3 = Label(Window_OnlineShop, text="NEXT GOOD",
font=("Arial Bold", 36), bg="#F0E68C")
OnlineShop_Lbl3.place(relx=0.45, rely=0.58, anchor="c")

OnlineShop_ButtonPreviousGood = Button(Window_OnlineShop, text="<--",
font=("Arial Bold", 40), bd=10,
                                background="#BDB76B",
state=PreviousGoodButtonState,
                                command=OnlineShop_ButtonPreviousGood_Clicked)

```

```

OnlineShop_ButtonPreviousGood.place(relx=0.15, rely=0.7, anchor="c")

OnlineShop_ButtonNextGood = Button(Window_OnlineShop, text="-->",
font=("Arial Bold", 40), bd=10,
background="#BDB76B", state=NextGoodButtonState,
command=OnlineShop_ButtonNextGood_Clicked)
OnlineShop_ButtonNextGood.place(relx=0.45, rely=0.7, anchor="c")

OnlineShop_Lbl_Basket = Label(Window_OnlineShop, text="BASKET:",
font=("Arial Bold", 36), bg="#F0E68C")
OnlineShop_Lbl_Basket.place(relx=0.8, rely=0.55, anchor="c")

OnlineShop_BasketInfoTxt = Text(Window_OnlineShop, height=7, width=35,
bd=5, font=("Times New Roman", 24))
OnlineShop_BasketInfoTxt.place(relx=0.79, rely=0.77, anchor="c")

for i in range(len(TextForBasket)):
    OnlineShop_BasketInfoTxt.insert(INSERT, TextForBasket[i])

Window_OnlineShop.mainloop()

def AccountInfo_ButtonGoShopping_Clicked():
    Window_AccountInfo.withdraw()
    Window_OnlineShop_PowerOn()

def AccountInfo_ButtonSignOut_Clicked():
    Window_AccountInfo.withdraw()
    Window_SignIn_PowerOn()

def AccountInfo_ButtonConfirmNewEmail_Clicked():
    if '@' in str(AccountInfo_NewEmail.get()):
        sql = f"UPDATE ACCOUNTS SET email =
'{str(AccountInfo_NewEmail.get())}' WHERE email = '{str(SignIn_Email.get())}'"
        cursor.execute(sql)
        conn.commit()
        messagebox.showinfo("Online Shop - Account Information",
            "Email changed successfully! You need to sign in again.")
        Window_AccountInfo.withdraw()
        Window_SignIn_PowerOn()
    else:

```

```

        messagebox.showinfo("Online Shop - Sign Up", f"Entered email
{str(AccountInfo_NewEmail.get())} is not valid!")

def AccountInfo_ButtonConfirmNewPassword_Clicked():
    if 7 < len((str(AccountInfo_NewPassword.get()))) < 17:
        sql = f"UPDATE ACCOUNTS SET password =
'{str(AccountInfo_NewPassword.get())}' WHERE email =
'{str(SignIn_Email.get())}'"
        cursor.execute(sql)
        conn.commit()
        messagebox.showinfo("Online Shop - Account Information",
                             "Password changed successfully! You need to sign in again")
        Window_AccountInfo.withdraw()
        Window_SignIn_PowerOn()
    else:
        messagebox.showinfo("Online Shop - Sign Up", f"Password must be from 8 to
16 characters!")

def Chiphered_Password(size):
    return '*' * size

def Window_AccountInfo_PowerOn():
    Window_AccountInfo.deiconify()
    Window_AccountInfo.geometry('1500x800')
    Window_AccountInfo.resizable(width=False, height=False)
    Window_AccountInfo["bg"] = "#F08080"
    Window_AccountInfo.title("Online Shop - Account Information")

    AccountInfo_Lbl1 = Label(Window_AccountInfo, text="ACCOUNT
INFORMATION", font=("Arial Bold", 50), bg="#F08080")
    AccountInfo_Lbl1.place(relx=0.3, rely=0.08, anchor="c")

    AccountInfo_ButtonGoShopping = Button(Window_AccountInfo, text="GO
SHOPPING", font=("Arial Bold", 24), bd=10,
                                         background="#DC143C",
command=AccountInfo_ButtonGoShopping_Clicked)
    AccountInfo_ButtonGoShopping.place(relx=0.7, rely=0.08, anchor="c")

    AccountInfo_ButtonSignOut = Button(Window_AccountInfo, text="SIGN OUT",
font=("Arial Bold", 24), bd=10,
                                         background="#DC143C",

```



```

command=AccountInfo_ButtonSignIn_Clicked)
AccountInfo_ButtonSignIn.place(relx=0.9, rely=0.08, anchor="c")

AccountInfo_Lbl2 = Label(Window_AccountInfo, text="Current EMAIL",
font=("Arial Bold", 50), bg="#F08080")
AccountInfo_Lbl2.place(relx=0.25, rely=0.3, anchor="c")

AccountInfo_Lbl3 = Label(Window_AccountInfo, text="Current PASSWORD",
font=("Arial Bold", 50), bg="#F08080")
AccountInfo_Lbl3.place(relx=0.7, rely=0.3, anchor="c")

AccountInfo_Lbl4 = Label(Window_AccountInfo, text=str(SignIn_Email.get()),
font=("Arial Bold", 36), bg="#F08080")
AccountInfo_Lbl4.place(relx=0.25, rely=0.4, anchor="c")

AccountInfo_Lbl5 = Label(Window_AccountInfo,
text=Chiphered_Password(len(str(SignIn_Password.get()))),
font=("Arial Bold", 36), bg="#F08080")
AccountInfo_Lbl5.place(relx=0.7, rely=0.4, anchor="c")

AccountInfo_Lbl6 = Label(Window_AccountInfo, text="Change EMAIL?",
font=("Arial Bold", 50), bg="#F08080")
AccountInfo_Lbl6.place(relx=0.25, rely=0.6, anchor="c")

AccountInfo_Lbl7 = Label(Window_AccountInfo, text="Change PASSWORD?",
font=("Arial Bold", 50), bg="#F08080")
AccountInfo_Lbl7.place(relx=0.7, rely=0.6, anchor="c")

AccountInfo_NewEmailTxt = Entry(Window_AccountInfo, width=20, bd=5,
font=("Arial Bold", 36),
textvariable=AccountInfo_NewEmail)
AccountInfo_NewEmailTxt.place(relx=0.25, rely=0.7, anchor="c")

AccountInfo_NewPasswordTxt = Entry(Window_AccountInfo, width=20, bd=5,
font=("Arial Bold", 36), show='*',
textvariable=AccountInfo_NewPassword)
AccountInfo_NewPasswordTxt.place(relx=0.7, rely=0.7, anchor="c")

AccountInfo_ButtonConfirmNewEmail = Button(Window_AccountInfo,
text="CONFIRM", font=("Arial Bold", 24), bd=10,
background="#DC143C",

```



```

command=AccountInfo_ButtonConfirmNewEmail_Clicked)
    AccountInfo_ButtonConfirmNewEmail.place(relx=0.25, rely=0.85, anchor="c")

    AccountInfo_ButtonConfirmNewPassword = Button(Window_AccountInfo,
text="CONFIRM", font=("Arial Bold", 24), bd=10,
                background="#DC143C",

command=AccountInfo_ButtonConfirmNewPassword_Clicked)
    AccountInfo_ButtonConfirmNewPassword.place(relx=0.7, rely=0.85, anchor="c")

    Window_AccountInfo.mainloop()

def PreparingOrder_ButtonGoShopping_Clicked():
    Window_PreparingOrder.withdraw()
    Window_OnlineShop_PowerOn()

def PreparingOrder_ButtonSignOut_Clicked():
    Window_PreparingOrder.withdraw()
    Window_SignIn_PowerOn()

def PreparingOrder_ButtonConfirmOrder_Clicked():
    if '+' in str(MobilePhone.get()) and len(str(MobilePhone.get())) == 13:
        messagebox.showinfo("Online Shop - Preparing Order", "Order successfully
prepared! We will contact you soon!")
        global CurrentOrder
        CurrentOrder = []
        global CurrentID
        CurrentID = 1
        Window_PreparingOrder.withdraw()
        Window_OnlineShop_PowerOn()
    else:
        messagebox.showinfo("Online Shop - Preparing Order", "Invalid mobile phone
entered!")

def Window_PreParingOrder_PowerOn():
    Window_PreparingOrder.deiconify()
    Window_PreparingOrder.geometry('1500x800')
    Window_PreparingOrder.resizable(width=False, height=False)
    Window_PreparingOrder["bg"] = "#87CEFA"
    Window_PreparingOrder.title("Online Shop - Preparing Order")

```

```

PreparingOrder_Lbl1 = Label(Window_PreparingOrder, text="PREPARING
ORDER", font=("Arial Bold", 50), bg="#87CEFA")
PreparingOrder_Lbl1.place(relx=0.25, rely=0.08, anchor="c")

PreparingOrder_ButtonGoShopping = Button(Window_PreparingOrder, text="GO
SHOPPING", font=("Arial Bold", 24), bd=10,
background="#1E90FF",
command=PreparingOrder_ButtonGoShopping_Clicked)
PreparingOrder_ButtonGoShopping.place(relx=0.7, rely=0.08, anchor="c")

PreparingOrder_ButtonSignOut = Button(Window_PreparingOrder, text="SIGN
OUT", font=("Arial Bold", 24), bd=10,
background="#1E90FF",
command=PreparingOrder_ButtonSignOut_Clicked)
PreparingOrder_ButtonSignOut.place(relx=0.9, rely=0.08, anchor="c")

PreparingOrder_Lbl2 = Label(Window_PreparingOrder, text="YOUR ORDER:",
font=("Arial Bold", 50), bg="#87CEFA")
PreparingOrder_Lbl2.place(relx=0.17, rely=0.25, anchor="c")

PreparingOrder_OrderInfoTxt = Text(Window_PreparingOrder, height=7,
width=59, bd=5, font=("Times New Roman", 24))
PreparingOrder_OrderInfoTxt.place(relx=0.66, rely=0.35, anchor="c")

TextForBasket = []
global CurrentOrder
k = 0
WholePrice = 0.0
for GoodSort in CurrentOrder:
    k += 1
    WholePrice += GoodSort[2] * GoodSort[3]
    TextForBasket.append(str(k) + ") " + GoodSort[1] + " ( " + "ID = " +
str(GoodSort[0]) + " ), " +
str(GoodSort[3]) + " items, whole price - " + str(GoodSort[2] *
GoodSort[3]) + "$\n")

TextForBasket.append("WHOLE SUM: " + str(WholePrice) + "$")

for i in range(len(TextForBasket)):
    PreparingOrder_OrderInfoTxt.insert(INSERT, TextForBasket[i])

```

```

    PreparingOrder_Lbl3 = Label(Window_PreparingOrder, text="Indicate your
contacts and we will accept your order:",
                                font=("Arial Bold", 36), bg="#87CEFA")
    PreparingOrder_Lbl3.place(relx=0.4, rely=0.6, anchor="c")

    PreparingOrder_Lbl4 = Label(Window_PreparingOrder, text="Mobile Phone:",
font=("Arial Bold", 36), bg="#87CEFA")
    PreparingOrder_Lbl4.place(relx=0.2, rely=0.7, anchor="c")

    PreparingOrder_MobilePhoneTxt = Entry(Window_PreparingOrder, width=13,
bd=5, font=("Arial Bold", 36),
                                textvariable=MobilePhone)
    PreparingOrder_MobilePhoneTxt.place(relx=0.438, rely=0.7, anchor="c")

    PreparingOrder_Lbl5 = Label(Window_PreparingOrder, text="Address:",
font=("Arial Bold", 36), bg="#87CEFA")
    PreparingOrder_Lbl5.place(relx=0.238, rely=0.8, anchor="c")

    PreparingOrder_AddressTxt = Text(Window_PreparingOrder, height=3, width=60,
bd=5, font=("Times New Roman", 24))
    PreparingOrder_AddressTxt.place(relx=0.64, rely=0.85, anchor="c")

    PreparingOrder_ButtonConfirmOrder = Button(Window_PreparingOrder,
text="CONFIRM ORDER", font=("Arial Bold", 24),
                                bd=10, background="#1E90FF",
command=PreparingOrder_ButtonConfirmOrder_Clicked)
    PreparingOrder_ButtonConfirmOrder.place(relx=0.15, rely=0.92, anchor="c")

    Window_PreparingOrder.mainloop()
# --- БЛОК РАБОТЫ С GUI ---

Window_SignIn_PowerOn() # Запуск со стартовой страницы

```

### 6.3. Скриншоты работы приложения

The image displays two screenshots of a web application interface. The top screenshot is the 'SIGN IN' page, which has a light green background. It features a title 'SIGN IN' at the top center. Below it, there are two input fields: 'Enter Email:' with the value 'test@mail.com' and 'Enter Password:' with the value '\*\*\*\*\*'. A small link 'Don't have account yet? Sign Up!' is positioned below the password field. At the bottom center is a green button labeled 'CONFIRM'. The bottom screenshot is the 'SIGN UP' page, which has a light purple background. It features a title 'SIGN UP' at the top center. Below it, there are three input fields: 'Enter Email:' with the value 'fpm.zhukovskps@bsu.by', 'Enter Password:' with the value '\*\*\*\*\*', and 'Confirm Password:' with the value '\*\*\*\*\*'. A small link 'Already have account? Sign in' is positioned below the password fields. At the bottom center is a purple button labeled 'CONFIRM'. Both screenshots show window control buttons (minimize, maximize, close) in the top right corner.

Online Shop - Sign In

## SIGN IN

Enter Email: test@mail.com

Enter Password: \*\*\*\*\*

[Don't have account yet? Sign Up!](#)

CONFIRM

Online Shop - Sign Up

## SIGN UP

Enter Email: fpm.zhukovskps@bsu.by

Enter Password: \*\*\*\*\*

Confirm Password: \*\*\*\*\*

[Already have account? Sign in](#)

CONFIRM

## ACCOUNT INFORMATION

GO SHOPPING

SIGN OUT

Current EMAIL

test@mail.com

Current PASSWORD

\*\*\*\*\*

Change EMAIL?

test@mail.ru

CONFIRM

Change PASSWORD?

\*\*\*\*\*

CONFIRM

## GOOD's SHOP

ACCOUNT INFORMATION

SIGN OUT

Good's ID	Good's Name	Price	Amount in stock
1	Ferrari	299999.99\$	12

Take some?



TAKE TO BASKET

PREVIOUS GOOD



NEXT GOOD



BASKET:

CONFIRM AND GO TO THE ORDER PREPARING

Online Shop - Catalog

GOOD's SHOP

ACCOUNT INFORMATION

SIGN OUT

Good's ID	Good's Name	Price	Amount in stock	Take some?
14	PS2	430.99\$	11	<div>8</div> <div>TAKE TO BASKET</div>

PREVIOUS GOOD

<--

NEXT GOOD

-->

CONFIRM AND GO TO THE ORDER PREPARING

BASKET:

1) Ferrari ( ID = 1 ), 2 items, whole price - 599999.98\$  
2) Geforce GTX 1080 Ti ( ID = 3 ), 4 items , whole price - 2399.96\$  
3) Guitar ( ID = 6 ), 14 items, whole price - 489.86\$  
4) Computer ( ID = 9 ), 3 items, whole price - 3722.9700000000003\$

Online Shop - Preparing Order

PREPARING ORDER

GO SHOPPING

SIGN OUT

YOUR ORDER:

1) Ferrari ( ID = 1 ), 2 items, whole price - 599999.98\$  
2) Geforce GTX 1080 Ti ( ID = 3 ), 4 items, whole price - 2399.96\$  
3) Guitar ( ID = 6 ), 14 items, whole price - 489.86\$  
4) Computer ( ID = 9 ), 3 items, whole price - 3722.9700000000003\$  
5) Book ( ID = 12 ), 51 items, whole price - 509.49\$  
6) PS2 ( ID = 14 ), 2 items, whole price - 861.98\$  
WHOLE SUM: 607984.2399999999\$

Indicate your contacts and we will accept your order:

Mobile Phone:

+375291405376

Address:

пр. Дзержинского, д. 141, кв. 231

CONFIRM ORDER

#### 6.4. Ссылка на все исходные файлы проекта (включая sql-файл)

Ссылка: <https://drive.google.com/drive/u/0/folders/1bzjVBcKMso3qoljFMfXoa8Fs-lf-7Le6>

Ссылка на GitHub с исходным кодом и БД:

[https://github.com/Shist/PrPS\\_Lab\\_6\\_MainApplication](https://github.com/Shist/PrPS_Lab_6_MainApplication)

## 7. Результаты тестирования системы

### 7.1. Unit testing

1. Подключение к Базе Данных через sqlite3

Итог: успешное подключение.

2. Правильность ввода количества того или иного товара:  $\text{Good.number} > 0$ .

Тестируемый класс: Good.

Входные данные	Ожидаемый результат	Результат	Тест прошел
6	true	true	Да
0	false	false	Да
-4	false	false	Да

3. Правильность ввода количества всех видов товаров в заказе:

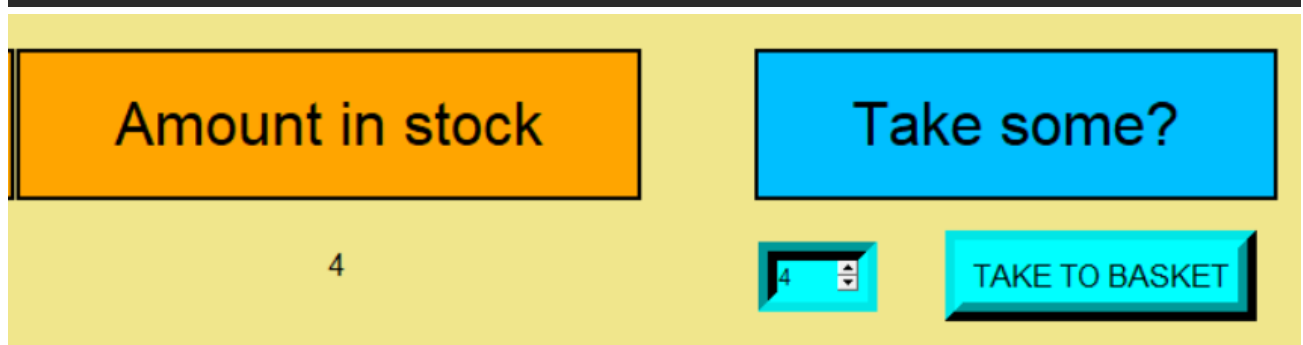
$\text{Order.GoodsCount} > 0$ .

Тестируемый класс: Order.

Входные данные	Ожидаемый результат	Результат	Тест прошел
9	true	true	Да
-5	false	false	Да
0	false	false	Да

Мне удалось полностью исключить ошибки в подобных тестах благодаря реализации объекта scrollbar таким образом, что выбрать можно не менее единицы товара и не более максимального значения, что есть на складе:

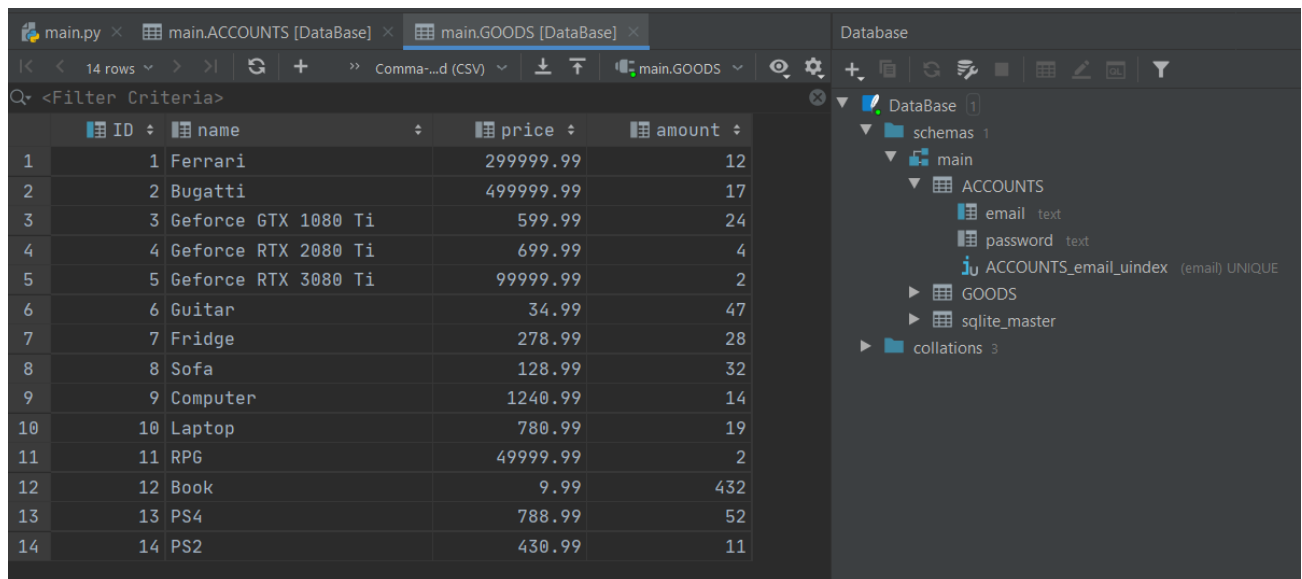
```
OnlineShop_SpinBox = Spinbox(Window_OnlineShop, from_=1, to=CurrentAmount, width=5, bg="#00FFFF", bd=10,
                              textvariable=HowManyWasTaken)
OnlineShop_SpinBox.place(relx=0.78, rely=0.45, anchor="c")
```



## 7.2. Integration testing

1. Пользователь заказал определенные товары, тем самым эти товары пришлось взять со склада: изменение в БД соответствующей таблицы.
2. Пользователь по каким-то причинам отменил свой заказ, таким образом какие-то товары снова придется предоставить на склад: изменение в БД соответствующей таблицы.
3. Корректное отображение на GoodsForm информации из БД.
4. Корректное отображение на OrderForm информации из БД.

Эти данные очень просто подкорректировать разработчикам системы благодаря интерфейсу среды разработки PyCharm, поддерживающую работу с БД типа SQLite:



ID	name	price	amount
1	Ferrari	299999.99	12
2	Bugatti	499999.99	17
3	Geforce GTX 1080 Ti	599.99	24
4	Geforce RTX 2080 Ti	699.99	4
5	Geforce RTX 3080 Ti	99999.99	2
6	Guitar	34.99	47
7	Fridge	278.99	28
8	Sofa	128.99	32
9	Computer	1240.99	14
10	Laptop	780.99	19
11	RPG	49999.99	2
12	Book	9.99	432
13	PS4	788.99	52
14	PS2	430.99	11

## 7.3. System testing

Для работы с системой пользователю необходимо авторизоваться. Для этого пользователь вводит логин и пароль. Система проверяет наличие данных в БД. Если данные корректные, то пользователю становится доступен весь функционал приложения, доступный данному типу пользователя.

Необходимо протестировать:

1. Корректность ввода логина и пароля.
2. При нажатии на каждую кнопку происходит определенное действие.
3. Данные, содержащиеся в БД, отображаются корректно и соответствуют запросу.



4. Корректно отображается текст приложения, кнопки и прочие элементы.

Тестирование всех этих аспектов видно на скриншотах работы приложения выше, а также на скриншотах Scenario testing.

#### 7.4. Usability testing

1. Корректное отображение всех кнопок, окон и прочих объектов на форме GoodsForm.

2. Корректное отображение всех кнопок, окон и прочих объектов на форме OrderForm.

3. Размер каждой кнопки не менее 10 px по длине.

4. Размер каждой кнопки не менее 1 px по ширине.

5. Возможность корректной работы кнопок при их нажатии.

Корректность работы кнопок можно наблюдать на скриншотах работы приложения.

#### 7.5. Acceptance testing

1. Вход в систему пользователя.

Возможность посмотреть информацию об аккаунте и о заказах. Возможно редактировать информацию о заказах или об аккаунте. Также возможность создавать новые заказы или отменять текущие.

Данные возможности реализованы в специальном окошке, работа которого была показана на одном из скриншотов:

Online Shop - Account Information

**ACCOUNT INFORMATION** **GO SHOPPING** **SIGN OUT**

**Current EMAIL**  
test@mail.com

**Current PASSWORD**  
\*\*\*\*\*

**Change EMAIL?** **Change PASSWORD?**

test@mail.ru \*\*\*\*\*

**CONFIRM** **CONFIRM**

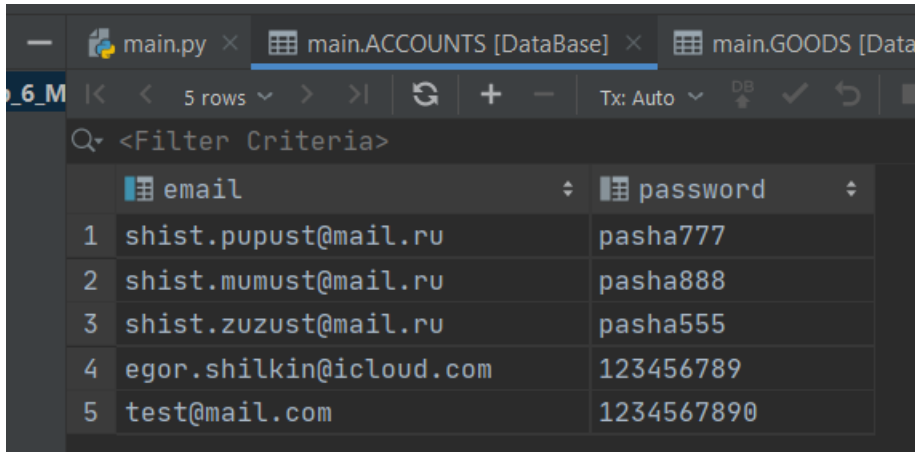
## 7.6. Scenario testing

**Сценарий событий для варианта использования «Войти в систему»:**

1. Введен корректный логин и корректный пароль. Авторизация прошла успешно.

email, пароль  
shist.pupust@mail.ru  
pasha777

Результат  
Вход в систему



	email	password
1	shist.pupust@mail.ru	pasha777
2	shist.mumust@mail.ru	pasha888
3	shist.zuzust@mail.ru	pasha555
4	egor.shilkin@icloud.com	123456789
5	test@mail.com	1234567890

Вход успешен:



**SIGN IN**

Enter Email: shist.pupust@mail.ru

Enter Password: \*\*\*\*\*

[Don't have account yet? Sign Up!](#)

**CONFIRM**

2. Введен некорректный (или пустой) логин и корректный пароль.  
Авторизация не прошла.

Логин, пароль	Результат
-	Неудача при входе в систему
2dk20kf2	
П0ьпм203а2а2б-3аб23аб	Неудача при входе в систему
2dk20kf2	
32а	Неудача при входе в систему
2dk20kf2	

3. Введен корректный логин и некорректный (или пустой) пароль.  
Авторизация не прошла.

Логин, пароль	Результат
Pavel0304	Неудача при входе в систему
20пъ2м23ъа23-аъ2-23ъз23аъ2	
Pavel0304	Неудача при входе в систему
290	

4. Введен некорректный (или пустой) логин и некорректный (или пустой) пароль. Авторизация не прошла.

Логин, пароль	Результат
В 423 t23t 23v23v23vt23 3	Неудача при входе в систему
32t23v23vtv2332v525v	
325	Неудача при входе в систему
dgs	
11111111111111111111	Неудача при входе в систему
222	
2	Неудача при входе в систему
Dddddddddddddddddd	

Тестирование вышеуказанных нюансов выполняется при регистрации, а уже при входе сверяется лишь с наличием данных в БД. Вот реализация данных проверок:

Проверка на валидность введённого email:

The screenshot shows a 'SIGN UP' form on a purple background. The form has three input fields: 'Enter Email:' containing 'spmdogmsdpgssdgsf', 'Enter Password:' containing '\*\*\*\*\*', and 'Confirm Password:' containing '\*\*\*\*\*'. Below the password fields is a 'CONFIRM' button. To the right of the 'Enter Email' field is a link that says 'Already have account? Sign In'. A modal dialog box titled 'Online Shop - Sign Up' is open, displaying an error message: 'Entered email spmdogmsdpgssdgsf is not valid!'. The dialog has an 'OK' button.

Проверка на слишком короткую длину пароля:

# SIGN UP

Enter Email:

Enter Password:

Confirm Password:

[Already have account? Sign In!](#)

Online Shop - Sign Up

Password must be from 8 to 16 characters!

Проверка на слишком длинную длину пароля:

Enter Email:

Enter Password:

Confirm Password:

[Already have account? Sign In!](#)

Online Shop - Sign Up

Password must be from 8 to 16 characters!

Проверка на совпадение паролей:

Enter Email:

Enter Password:

Confirm Password:

[Already have account? Sign In!](#)

**CONFIRM**

Online Shop - Sign Up  
Passwords do not match!  
OK

***Сценарий событий для варианта использования «Действие»:***

1. При выборе действия «сделать заказ» все выбранные товары доступны для продажи. Действие прошло успешно.

Доступность товаров	Результат
True	Успешное действие
True	
True	

2. При выборе действия «выбрать количество товаров» введено некорректное значение. Действие не прошло.

Количество товаров какого-то вида	Результат
-4	Неудачное действие
0	Неудачное действие
-2	Неудачное действие

3. При выборе действия «указать общее количество товаров» в заказе выбрано некорректное значение

Общее количество товаров в заказе	Результат
-5	Неудачное действие
0	Неудачное действие

5. При выборе действия «Сделать заказ» введен некорректный мобильный телефон.

Номер помещения, код  
подразделения  
109aa2ь9012ьa021

Результат  
Неудачное действие

Все вышеуказанные нюансы соблюдаются в реализации программы, однако для проверки на валидность введённого мобильного телефона была реализована дополнительная проверка:

Indicate your contacts and we will accept your order:

Mobile Phone:

Address:

Online Shop - Preparing Order

Invalid mobile phone entered!

OK