Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт: «Компьютерные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Базы данных»

**Курсовой проект**

**Тема: «Сервис доставки и отслеживания заказов»**

|  |  |
| --- | --- |
| Cтуденты: | Голов А.Ю.  Знай А.О.  Постнов А.В.  Яруллин А.Р. |
| Группа: | М8О-301Б-21 |
| Преподаватель: | Киндинова В.В. |
| Дата: |  |
| Оценка: |  |

Москва 2023

Оглавление

[**Техническое задание** 3](#_Toc89450499)

[Постановка задачи 3](#_Toc89450500)

[Требования по функциональности 3](#_Toc89450501)

[Требования по интерфейсу 3](#_Toc89450502)

[Инструменты 4](#_Toc89450503)

**База** [**данных**](#_Toc89450506) 5

Модель базы данных5

Схема базы данных [6](#_Toc89450508)

**Хэндлеры серверной части приложения**11

**Графический интерфейс** 12

Регистрация пользователя и курьера 12

Регистрация поставшика 14

Регистрация администратора 16

Аутентификация 17

Отображение доступных товаров 18

Добавление товара поставщиком 19

Удаление товара администратором 21

# Техническое задание

## Постановка задачи

Разработать прикладное программное обеспечение для создания и отслеживания заказов. Приложение должно реализовывать работу с основополагающими для выбранной сферы сущностями: продукт, заказ, пользователь, курьер, поставщик, администратор. В данном случае все четыре роли подразумевают хранение личной информации о каждой сущности.

## Требования по функциональности

Система должна предоставлять следующую функциональность:

1. Добавление данных в базу данных в следующих случаях:

- Регистрация роли;

- Создание заказа пользователем;

- Создание товара магазином;

1. Обновление данных базы данных:

- Изменение статуса заказа и времени его доставки;

1. Просмотр данных из базы данных:

- Заказы с вакантной ролью курьера;

- Личные данные пользователей;

- Доступные для заказа товары;

1. Удаление данных из базы данных:

- Удаление товара администратором;

## Требования по интерфейсу

Программа должна предоставлять для пользователя возможность лёгкого доступа к данным (на интуитивном уровне). Для этого у программы должен быть развитый графический интерфейс с командами:

1. Регистрация;
2. Аутентификация;
3. Наполнение корзины;
4. Создание заказа;
5. Взятие заказа курьером;
6. Изменение статуса и завершение заказа курьером;
7. Создание товара с загрузкой изображения;
8. Удаление товара;

Графический интерфейс данного приложения разработан с учетом различных ролей пользователей, обеспечивая удобство и эффективность взаимодействия с системой. Клиентский интерфейс предоставляет пользователям интуитивно понятные элементы управления для поиска, просмотра и заказа товаров. Центральная часть экрана отведена для отображения каталога товаров, а функционал корзины и истории заказов легко доступен через навигационные элементы.

Интерфейс курьера ориентирован на оптимизацию процесса доставки. Курьер может видеть список заказов, их статус и подробности, а также получать маршрут и необходимую информацию о клиенте. Панель управления позволяет курьеру отмечать выполненные доставки и вносить изменения в статус заказов.

Администраторский интерфейс обладает расширенными функциями, включая возможность управления товарами и пользователями. Администратор может удалять товары, следить за статистикой продаж, а также регулировать доступ и права пользователей. Гибкий и понятный функционал позволяет администратору эффективно управлять всеми аспектами приложения.

Интерфейс поставщика спроектирован для удобного добавления новых товаров в систему. Поставщик может легко вводить информацию о продуктах, указывать цены, загружать изображения и следить за актуальным состоянием своего ассортимента. Этот интерфейс позволяет поставщикам эффективно взаимодействовать с платформой, обеспечивая актуальность и разнообразие предлагаемых товаров.

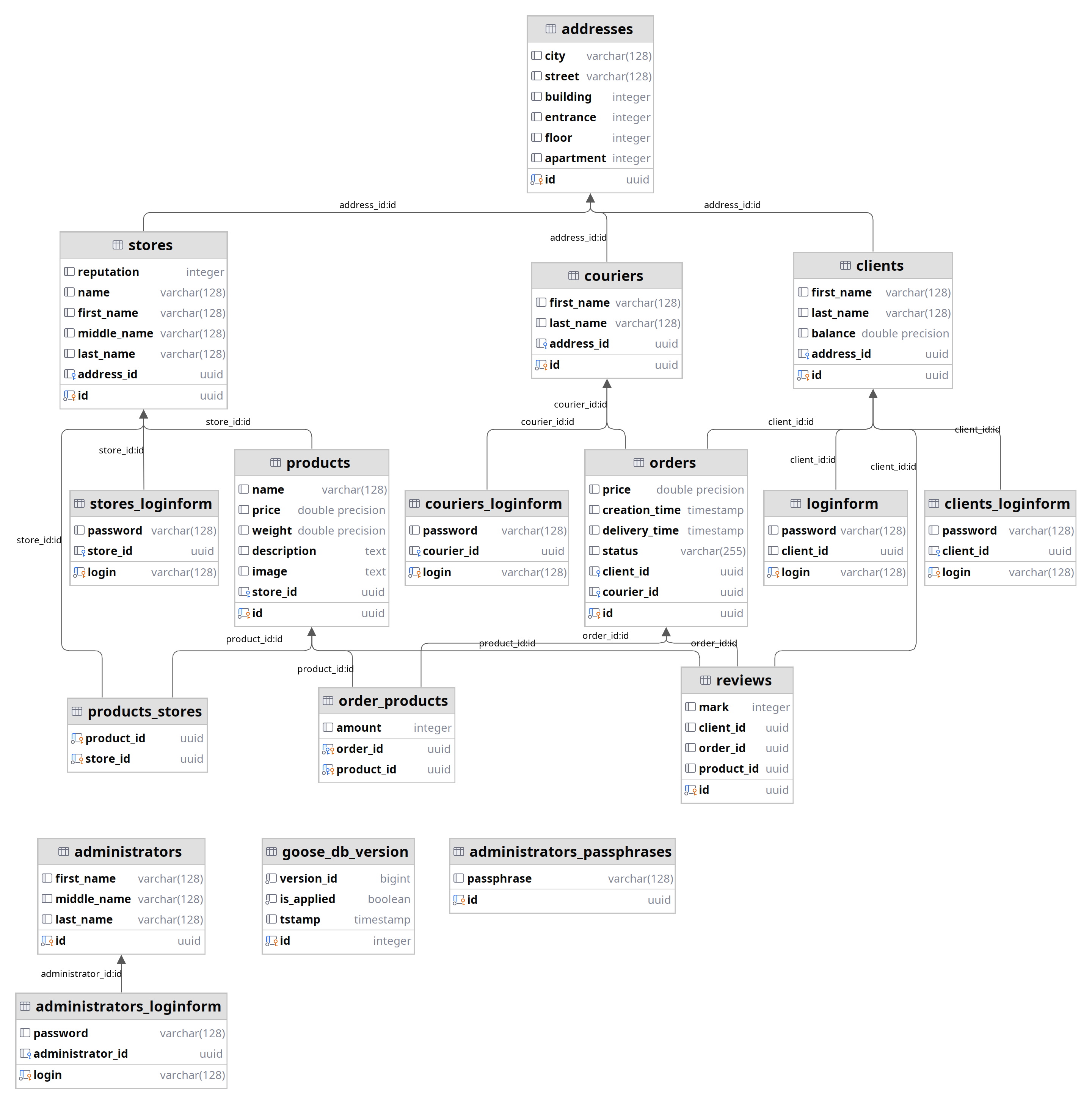
## Инструменты

СУБД — PostgreSQL;

Графический интерфейс — JavaScript, HTML, CSS, npm;

Серверная часть приложения — Golang REST API.

# Модель базы данных



# Схема базы данных в PostgreSQL

## 

## CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE IF NOT EXISTS addresses

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

city varchar(128),

street varchar(128),

building int,

entrance int,

floor int,

apartment int

);

CREATE TABLE IF NOT EXISTS clients

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

first\_name varchar(128),

last\_name varchar(128),

balance float,

address\_id UUID REFERENCES addresses (id)

);

CREATE TABLE IF NOT EXISTS clients\_loginform

(

login varchar(128) PRIMARY KEY,

password varchar(128),

client\_id UUID REFERENCES clients (id)

);

CREATE TABLE IF NOT EXISTS couriers

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

first\_name varchar(128),

last\_name varchar(128),

address\_id UUID REFERENCES addresses (id)

);

CREATE TABLE IF NOT EXISTS couriers\_loginform

(

login varchar(128) PRIMARY KEY,

password varchar(128),

courier\_id UUID REFERENCES couriers (id)

);

CREATE TABLE IF NOT EXISTS administrators

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

first\_name varchar(128),

middle\_name varchar(128),

last\_name varchar(128)

);

CREATE TABLE IF NOT EXISTS administrators\_loginform

(

login varchar(128) PRIMARY KEY,

password varchar(128),

administrator\_id UUID REFERENCES administrators (id)

);

CREATE TABLE IF NOT EXISTS administrators\_passphrases

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

passphrase varchar(128) -- Кодовое слово для регистрации администратора

);

CREATE TABLE IF NOT EXISTS stores

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

reputation int, -- Считается как-нибудь

name varchar(128), -- Имя магазина

first\_name varchar(128), -- Данные юрлица-представителя

middle\_name varchar(128),

last\_name varchar(128),

address\_id UUID REFERENCES addresses (id) -- Юридический адрес

);

CREATE TABLE IF NOT EXISTS stores\_loginform

(

login varchar(128) PRIMARY KEY,

password varchar(128),

store\_id UUID REFERENCES stores (id)

);

CREATE TABLE IF NOT EXISTS products

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

name varchar(128),

price float CHECK (price > 0),

weight float,

description text,

image text,

store\_id UUID REFERENCES stores (id)

);

CREATE TABLE IF NOT EXISTS orders

(

id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,

price float,

creation\_time timestamp,

delivery\_time timestamp,

status varchar(255),

client\_id UUID REFERENCES clients (id),

courier\_id UUID REFERENCES couriers (id)

);

CREATE TABLE IF NOT EXISTS order\_products

(

amount int,

order\_id UUID REFERENCES orders (id),

product\_id UUID REFERENCES products (id) ON DELETE CASCADE,

PRIMARY KEY (order\_id, product\_id)

);

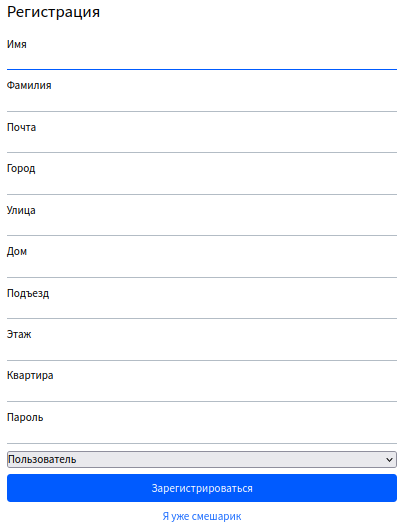
# Хэндлеры серверной части приложения

# 

# Графический интерфейс приложения

Для наглядности, сохраняя компактность будут приведены снимки экрана, содержащие элементы графического интерфейса и используемый SQL-запрос, опуская код серверной части приложения.

**Регистрация пользователя и курьера.**



checkLoginQuery = `

SELECT login, password, client\_id

FROM clients\_loginform WHERE login=$1

`

insertAddressQuery = `

INSERT INTO addresses(city, street, building, entrance, floor, apartment)

VALUES ($1, $2, $3, $4, $5, $6) RETURNING id

`

insertClientQuery = `

INSERT INTO clients(first\_name, last\_name, balance, address\_id)

VALUES($1, $2, $3, $4) RETURNING id

`

insertLoginFormQuery = `

INSERT INTO clients\_loginform(login, password, client\_id)

VALUES ($1, $2, $3)

`

getClientQuery = `

SELECT first\_name, last\_name, balance, address\_id

FROM clients

WHERE id=$1

`

getAddressQuery = `

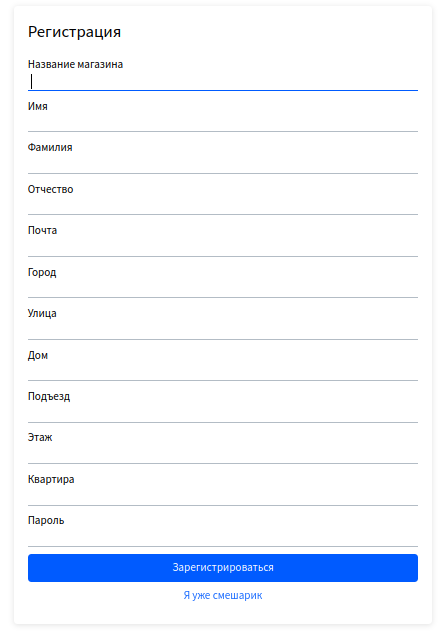
SELECT city, street, building, entrance, floor, apartment

FROM addresses

WHERE id=$1

`

**Регистрация поставщика**



insertAddressQuery = `

INSERT INTO addresses(city, street, building, entrance, floor, apartment)

VALUES ($1, $2, $3, $4, $5, $6) RETURNING id

`

insertStoreQuery = `

INSERT INTO stores(name, first\_name, middle\_name, last\_name, address\_id)

VALUES($1, $2, $3, $4, $5) RETURNING id

`

insertLoginFormQuery = `

INSERT INTO stores\_loginform(login, password, store\_id)

VALUES ($1, $2, $3)

`

checkLoginQuery = `

SELECT login, password, store\_id

FROM stores\_loginform

WHERE login=$1

`

selectStoreQuery = `

SELECT name, first\_name, middle\_name, last\_name, address\_id

FROM stores

WHERE id=$1

`

selectAddressQuery = `

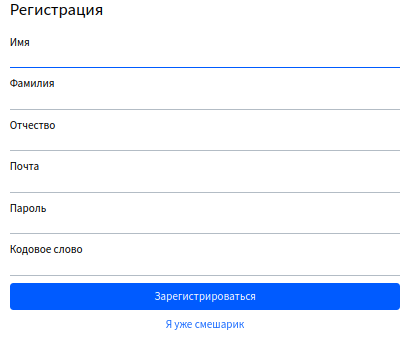
SELECT city, street, building, entrance, floor, apartment

FROM addresses

WHERE id=$1

`

**Регистрация администратора**



insertAdminQuery = `

INSERT INTO administrators(first\_name, middle\_name, last\_name)

VALUES ($1, $2, $3) RETURNING id

`

insertLoginFormQuery = `

INSERT INTO administrators\_loginform(login, password, administrator\_id)

VALUES ($1, $2, $3)

`

selectLoginFormQuery = `

SELECT login, password, administrator\_id

FROM administrators\_loginform

WHERE login=$1

`

selectAdminQuery = `

SELECT first\_name, middle\_name, last\_name

FROM administrators

WHERE id=$1

`

selectPassPhraseQuery = `

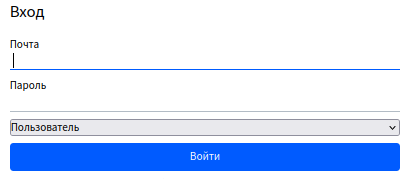
SELECT id

FROM administrators\_passphrases

WHERE passphrase=$1

`

**Аутентификация (для всех ролей)**



selectLoginFormQuery = `

SELECT login, password, administrator\_id

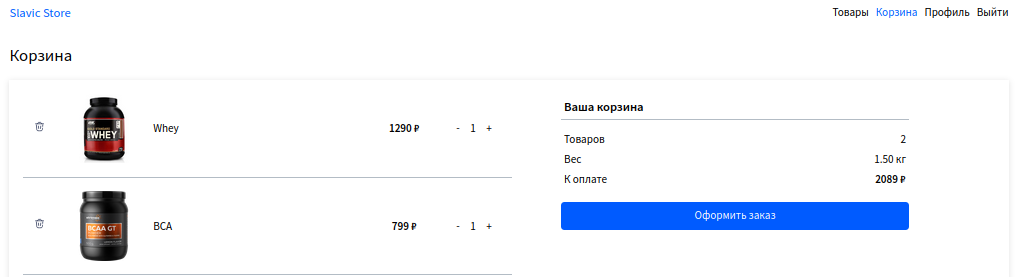
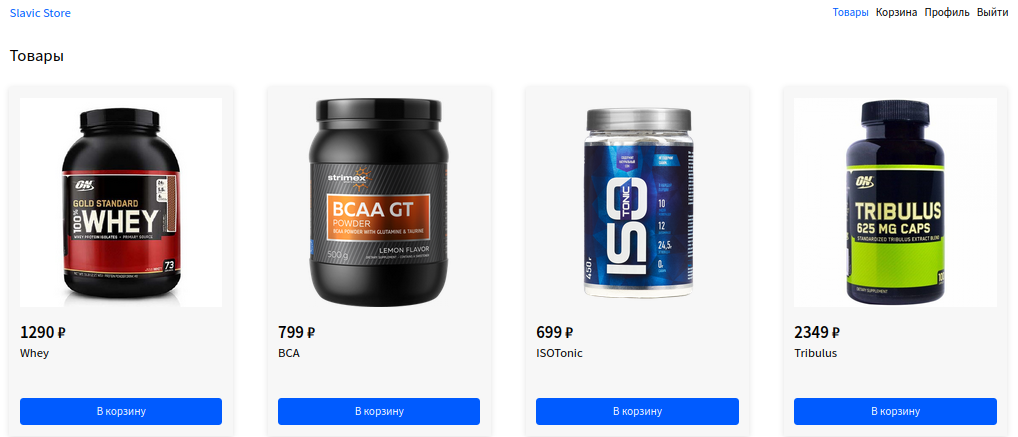
FROM administrators\_loginform

WHERE login=$1

`

Запросы для выборки логина, пароля и id сущности идентичны для всех ролей.

**Доступные для заказа товары и корзина пользователя**

getProductsQuery = `

SELECT id, name, price, weight, description, image

FROM products

`

getProductByID = `

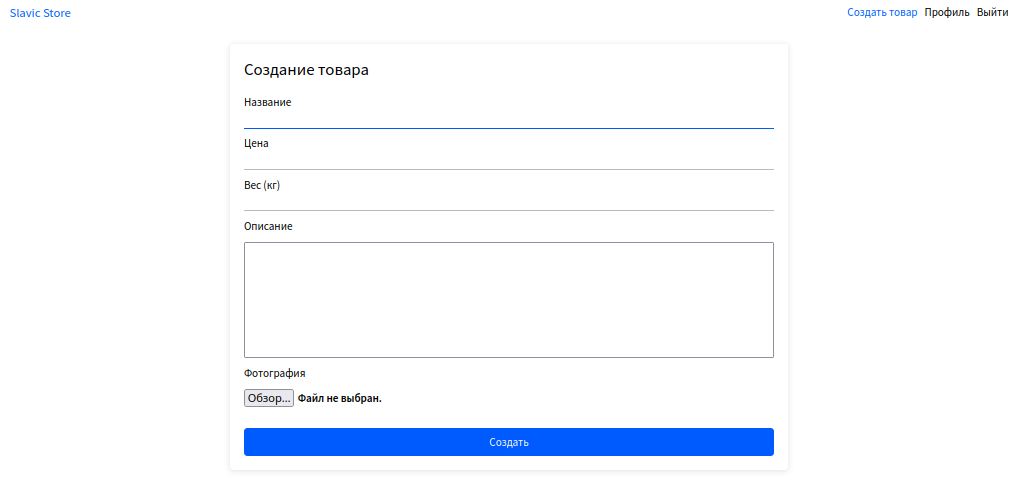
SELECT id, name, price, weight, description, image

FROM products

WHERE id=$1

`

**Создание товара поставщиком**

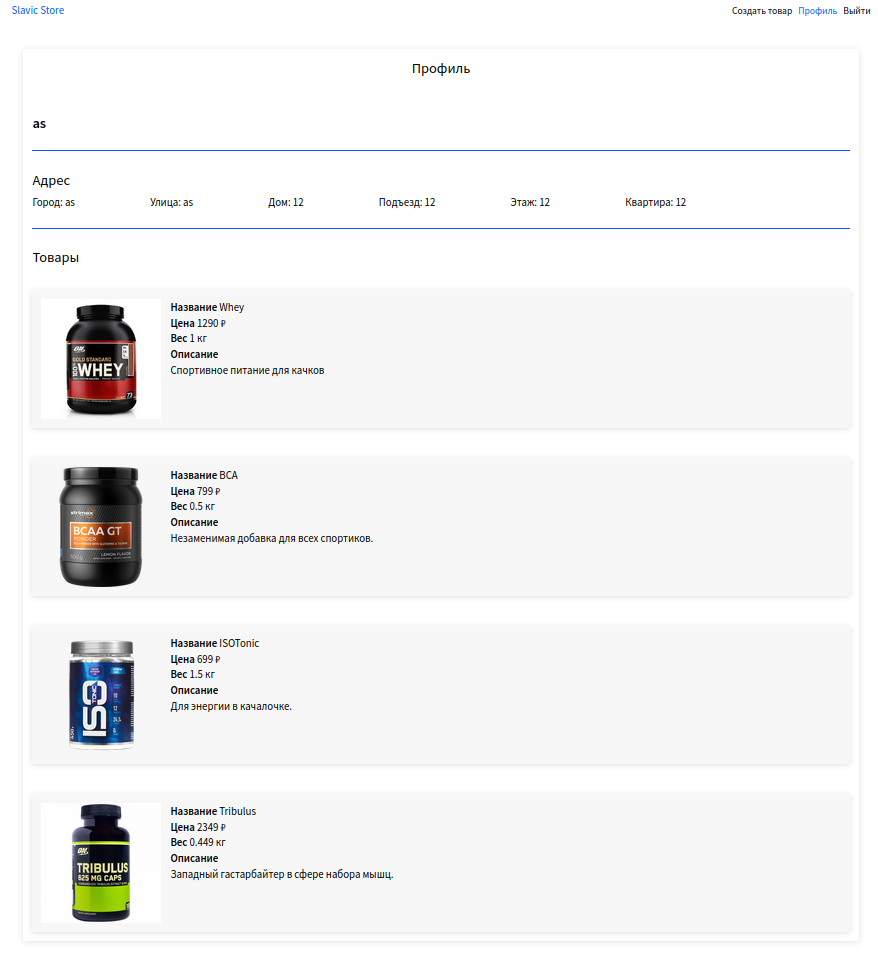
insertProductQuery = `

INSERT INTO products(name, price, weight, description, image, store\_id)

VALUES ($1, $2, $3, $4, $5, $6)RETURNING id

`

**Отображение товаров в профиле поставщика**

selectProductsByStore = `

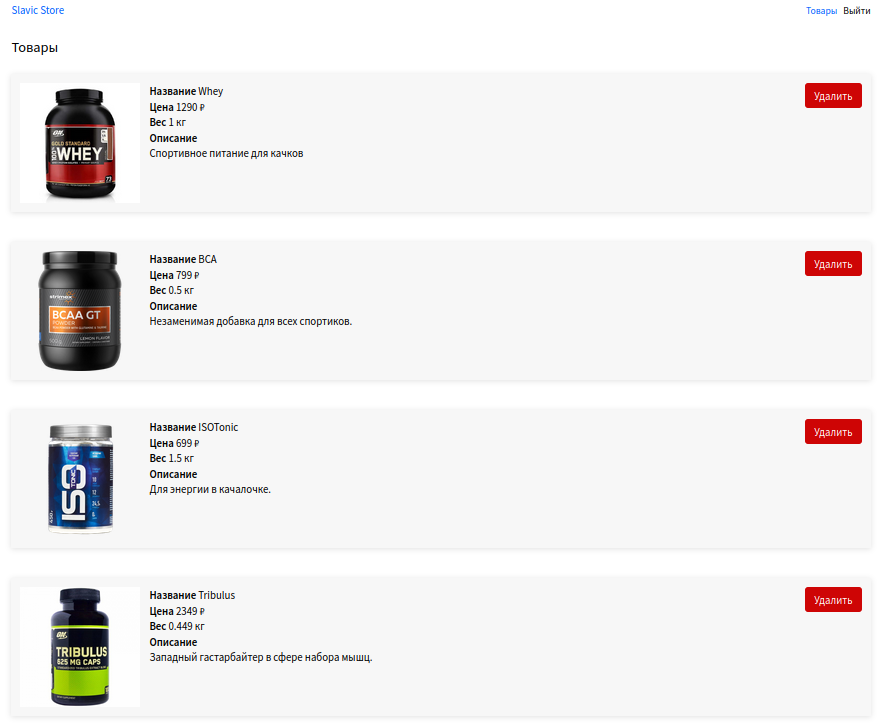
SELECT id, name, price, weight, description, image

FROM products

WHERE store\_id=$1

`

**Удаление товара администратором**

deleteProductById = `

DELETE FROM products

WHERE id=$1

`

Здесь важно отметить, что при удалении товара удалятся и все заказы, содержащие этот товар, это реализовано конструкцией ON DELETE CONSTRAINT;