**Московский государственный технический университет им. Н.Э. Баумана**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Утверждаю: |  |
| Галкин В.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Курсовая работа по дисциплине

«Сетевые технологии»

**«Сервер заказов с хранением видео»**

Пояснительная записка

(вид документа)

писчая бумага

(вид носителя)

7

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты групп ИУ5-61Б |  |
| Мартынова П.В.  Агличеев М.С. | "\_\_\_"\_\_\_\_\_\_\_\_\_2022 г. |

Москва - 2022

Оглавление

[**1.** **Введение** 2](#_Toc101398998)

[**2.** **Требования к программе** 2](#_Toc101398999)

[**3.** **Определение структуры программного продукта** 2](#_Toc101399000)

[**4.** **Backend (сервер).** 2](#_Toc101399001)

[**5.** **Frontend.** 3](#_Toc101399002)

1. **Введение**

Данное WEB-приложение (программа), выполненное в рамках курсовой работы по предмету «Сетевые технологии», предназначено для обеспечения функционирования интернет-магазина с возможностью хранения видео.

1. **Требования к программе**

К программе предъявляются следующие требования. Программа должна:

1. Обеспечить авторизацию пользователей
2. Предусмотреть разделение пользователей на администраторов и обычных пользователей
3. Обеспечить просмотр каталога товаров
4. Предусмотреть интерфейс администраторов для добавления элементов в API
5. Обеспечить возможность добавления товара в избранное и корзину
6. Обеспечить возможность оформления заказа
7. Обеспечить возможность хранения не только изображений, но и видео на странице товара
8. **Определение структуры программного продукта**

При взаимодействии пользователя с WEB-приложением выделяют 2 уровня – клиент и сервер.

- Клиент обеспечивает взаимодействие с пользователем: приём пользовательского ввода и вывод информации, полученной от сервера. Он индивидуален для каждого пользователя.

- Сервер на постоянной основе хранит в себе необходимую пользователям информацию, обрабатывает её и записывает/выдаёт по запросу. Он общий для всех пользователей (технологии вроде CDN не использовались в данной работе).

1. **Backend (сервер).**

Сервер выполнен на языке Go с использованием микросервисной архитектуры. Взаимодействие между микросервисами реализовано посредством gRPC. Для хранения пользовательских данных используется СУБД PostgreSQL, для проксирования запросов от клиента к микросервисам – Nginx.

Микросервисы можно условно разделить на несколько групп:

- Роутинг – принимает пользовательский запрос, парсит содержащиеся в нём данные и, исходя из этих данных, передаёт информацию о запросе другим микросервисам

- Авторизация – занимается обработкой входа/выхода пользователя из системы, проверкой, вошёл ли уже пользователь в систему, а также регистрацией и сменой авторизационных данных (логина и пароля)

- Пользователи – обрабатывает запросы на получение и изменение не-авторизационных пользовательских данных (имени, аватара и т.д.).

- Магазины – обрабатывает запросы на получение и изменение данных о магазинах, например, названии или местоположении

- Товары – предназначен для получения данных о товаре: цены, продавца и т.д. Частично обращается за информацией к микросервису магазинов.

Все эти микросервисы (с помощью проксирования) могут быть теоретически неограниченно дублированны и масштабируемы.

Между микросервисом роутинга и клиентом располагается «слой» Nginx-проксирования, который обеспечивает 2 функции – отдачу файлов клиентского приложения при первом заходе пользователя на сайт и проксирование запросов от клиента на микросервис роутинга с поддержкой HTTPS. SSL-сертификаты, необходимые для установления HTTPS-соединения, генерируются с помощью сервиса Certbot, который поддерживает генерацию пары секретного и публичного SSL-ключа с использованием исключительно файла конфигурации Nginx.

Из-за не очень большой нагрузки на сайт, оптимально в качестве СУБД использовать только PostgreSQL. В случае увеличения нагрузки, для хранения пользовательских сессий (обращение к которым происходит практически при каждом запросе пользователя) можно использовать NoSQL решения, вроде Redis.

Мультимедиа (изображения и видео) хранятся в S3-хранилище, общедоступном для скачивания, благодаря чему в БД требуется хранить только ссылки на них, которые отдаются клиентской части приложения в случае необходимости. Названия файлов в хранилище генерируются случайным образом и сортируются по дате для простоты ручного нахождения и удаления и предотвращения совпадений этих названий. Для отправки файлов в Golang есть пакет aws-sdk-go-2.

Развёртывание микросервисов выполняется с помощью Docker, что позволяет упростить периодическое развёртывание сервера при обновлениях до простого перезапуска docker-контейнеров с перестройкой образов.

Для роутинга в микросервисе роутинга используется пакет gorilla/mux. Он позволяет автоматически получать из запросов query-параметры а также создавать группы запросов по необходимости наличия/отсутствия авторизации. Сама эе проверка авторизованности происходит с помощью middleware, обращающегося к микросервису авторизации.

1. **Frontend.**

Пользовательский интерфейс выполнен в среде MS Visual Studio Code на языке JavaScript. При его разработке учитывались рекомендации по простоте, удобству и функциональности интерфейса. При разработке приложения использовалась библиотека React — библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через инкапсулированные единицы (называются компонентами), которые сохраняют состояние и генерируют элементы пользовательского интерфейса. Приложение создавалось с помощью Create React App (CRA) - интерфейса командной строки (command line interface, CLI), который позволяет создавать предварительно настроенные React-проекты. Предварительная настройка включает в себя, но не исчерпывается, настройкой Webpack и Babel.

Так как JavaScript не является типизированным языком, в проекте также использовался язык программирования TypeScript. Преимущества использования TypeScript состоят в том, что строгая типизация значительно сокращает количество возможных ошибок, удобно интегрируется с IDE, расширяет возможности JavaScript в ООП (TypeScript позволяет использовать наследование, абстрактные классы и многое другое), компилятор TS позволяет преобразовывать код в JS-код более старых стандартов языка, что повышает поддержку браузерами, а также повышает наглядность и документируемость кода.

Для управления состоянием в проекте используется библиотека mobx. Основная идея состоит в том, что состояние - это сердце каждого приложения и нет более быстрого способа создания забагованого, неуправляемого приложения, как отсутствие консистентности состояния. Или состояние, которое несогласованно с локальными переменными вокруг. Поэтому множество решений по управлению состоянием пытаются ограничить способы, которыми можно его изменять, например сделать состояние неизменяемым. Но это порождает новые проблемы, данные нуждаются в нормализации, нет гарантии ссылочной целостности и становится почти невозможно использовать такие мощные концепты как прототипы. MobX позволяет сделать управление состоянием вновь простым, вернувшись к корню проблемы: он делает невозможным инконсистентность состояния. Стратегия достижения этого довольно проста: убедится что, все что может быть вынуто из состояния, будет вынуто.

При переходе по ссылке <https://gears4us.ru/> в браузере открывается главная страница сайта.

Для навигации используется библиотека react-router-dom. При нажатии на кнопку с надписью «Каталог» открывается страница с каталогом товаров, отсортированных по категориям (рис. 1).

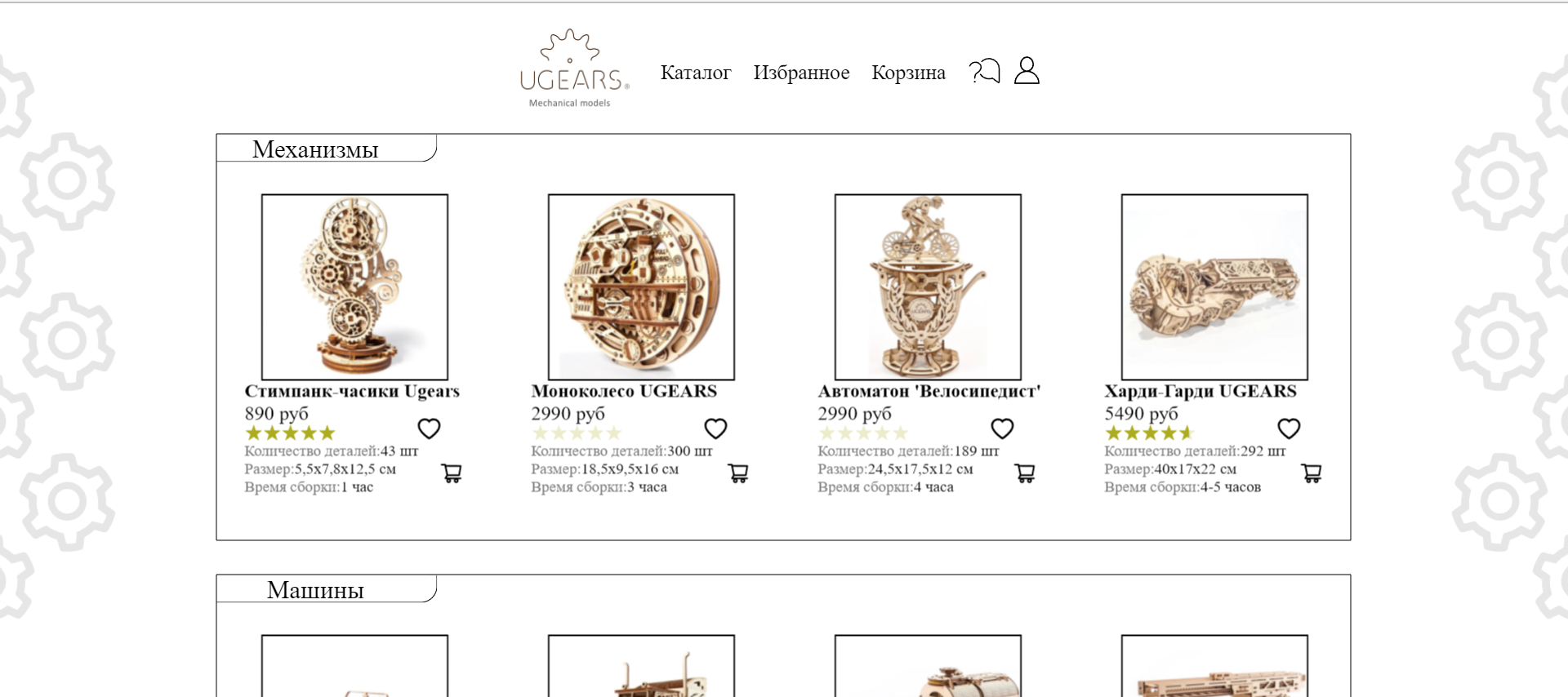


Рисунок 1- Каталог товаров

При нажатии на название товара, пользователь может перейти на страницу товара (рис. 2). Для получения данных о товаре со стороны frontend отравляется get-запрос, который содержит id продукта, id берется из адреса страницы с помощью хука useParams.

Для отображения картинок используется свайпер: при нажатии на картинку товара и перемещении курсора вправо или влево (свайпе) картинка меняется на следующую или предыдущую. Также выбор просматриваемой картинки может быть осуществлен с помощью миниатюр слева от просматриваемой картинки, для навигации используются стрелки. Чтобы перейти к просмотру видео, необходимо нажать на знак воспроизведения видео, который находится ниже миниатюр. Видео откроется вместо просматриваемой до этого картинки. В качестве слайдера был выбран Swiper, так как Swiper написан на чистом JavaScript, не требует подключения дополнительных библиотек, таких как jQuery, что делает его быстрее и легче.

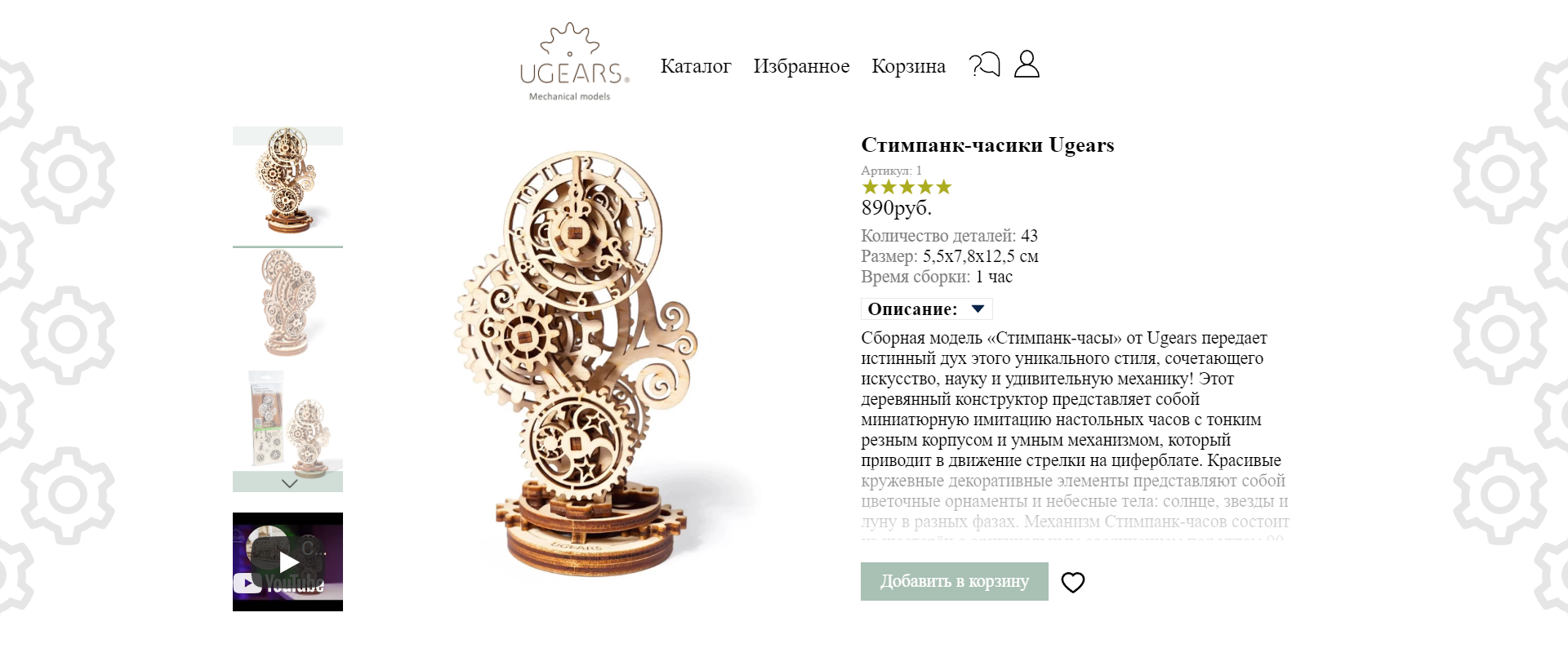


Рисунок 2. Страница товара

При добавлении товара в корзину на сервер отправляется put-запрос на cart/add/{productID}, после обработки запроса, сервер возвращает статус, если он равен 204, то данные о корзине обновляются, и пользователь может увидеть добавленный товар в корзине. При удалении товара из корзины на сервер отправляется put-запрос на cart/remove/{productID}. Аналогичные действия происходят при добавлении товара в избранное или удалении его оттуда.

Для того, чтобы оформить заказ, необходимо нажать на кнопку с надписью «Корзина» в шапке сайта. Кроме продуктов внутри корзины откроется форма с данными о заказе (рис. 5), пользователю необходимо заполнить следующие поля: адрес доставки, способ оплаты и отметить галочкой поле «звонок менеджера», если это требуется. При добавлении товара в корзину, а также в самой корзине можно указать количество товара (рис. 5). После оформления заказа у пользователя появится доступ к данным: дата и время заказа, состав заказаи статус заказа.

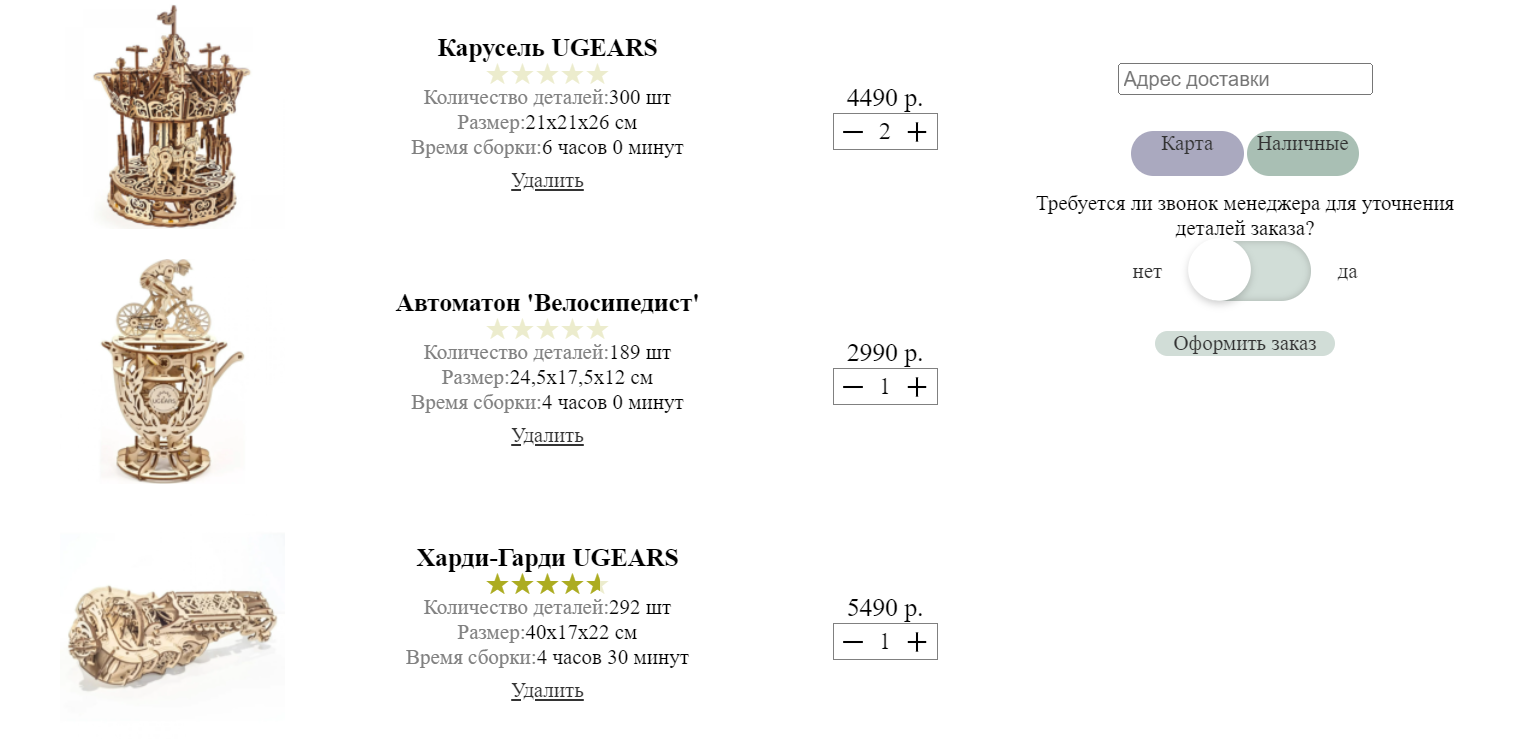


Рисунок 5 - Форма с заказом.

При нажатии пользователем на кнопку «оформить заказ», данные, введенные пользователем, отправляются на сервер в теле POST-запроса на cart/finish, после чего, при успешном результате, пользователю отправляются данные о сформированном заказе. Если пользователь указывал, что требуется звонок менеджера, то эти данные также отправляются менеджеру с пометкой о звонке.

Для того, чтобы совершать вышеперечисленные действия на сайте, пользователю необходимо авторизироваться. Для этого необходимо нажать на иконку с изображением человека в шапке сайта, после чего откроется форма для авторизации (рис. 5). Для того, чтобы авторизироваться, необходимо ввести логин и пароль и нажать на кнопку «отправить», после чего на сервер отправляется несколько запросов: post-запрос для отправки данных на сервер, get-запрос для определения статуса авторизации и get-запрос для получения данных об авторизированном пользователе.

В случае, если на сервере нет данных о пользователе, ему необходимо зарегистрироваться. Для этого нужно нажать на кнопку «Зарегистрироваться», после чего откроется форма регистрации (рис. 6).

Если авторизированный пользователь обладает статусом администратора или менеджера, ему открывается доступ к редактированию и удалению товаров.

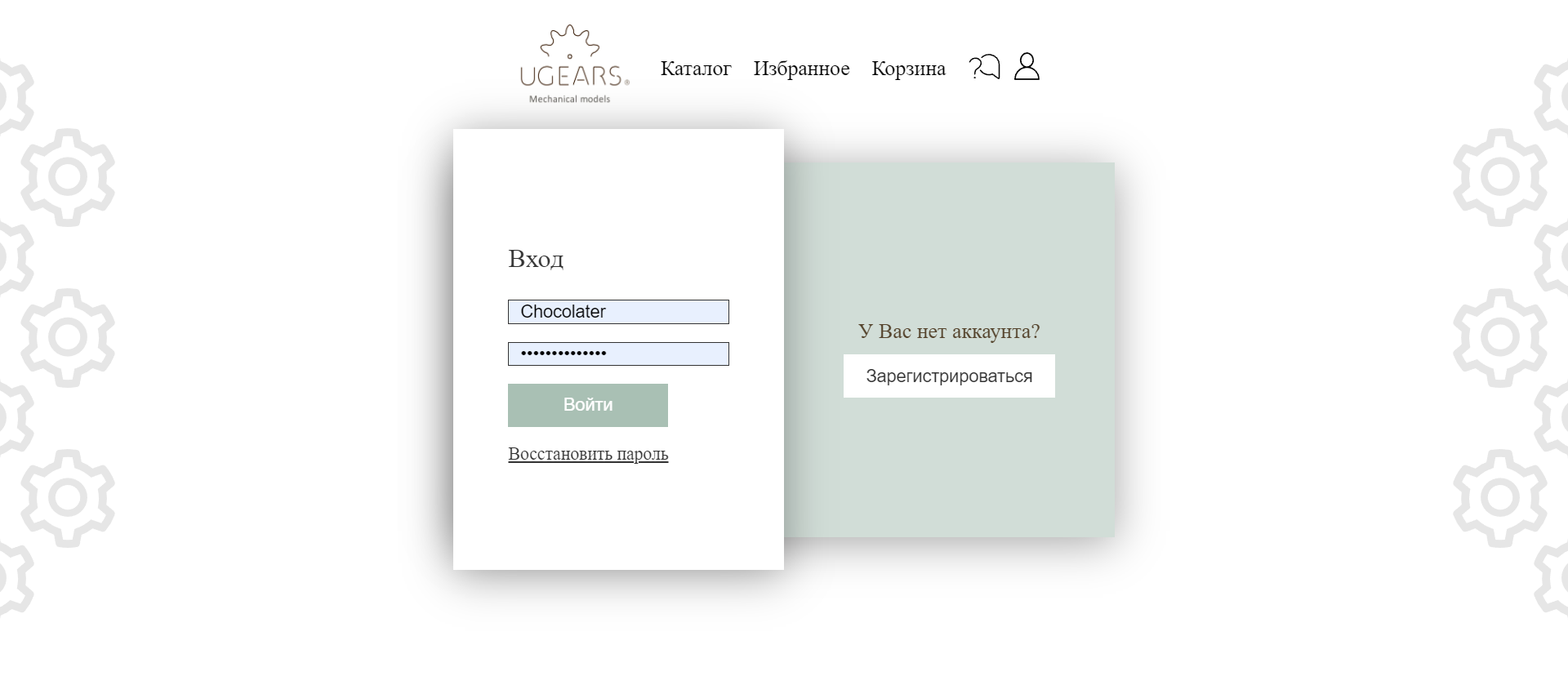


Рисунок 5. – Авторизация пользователя

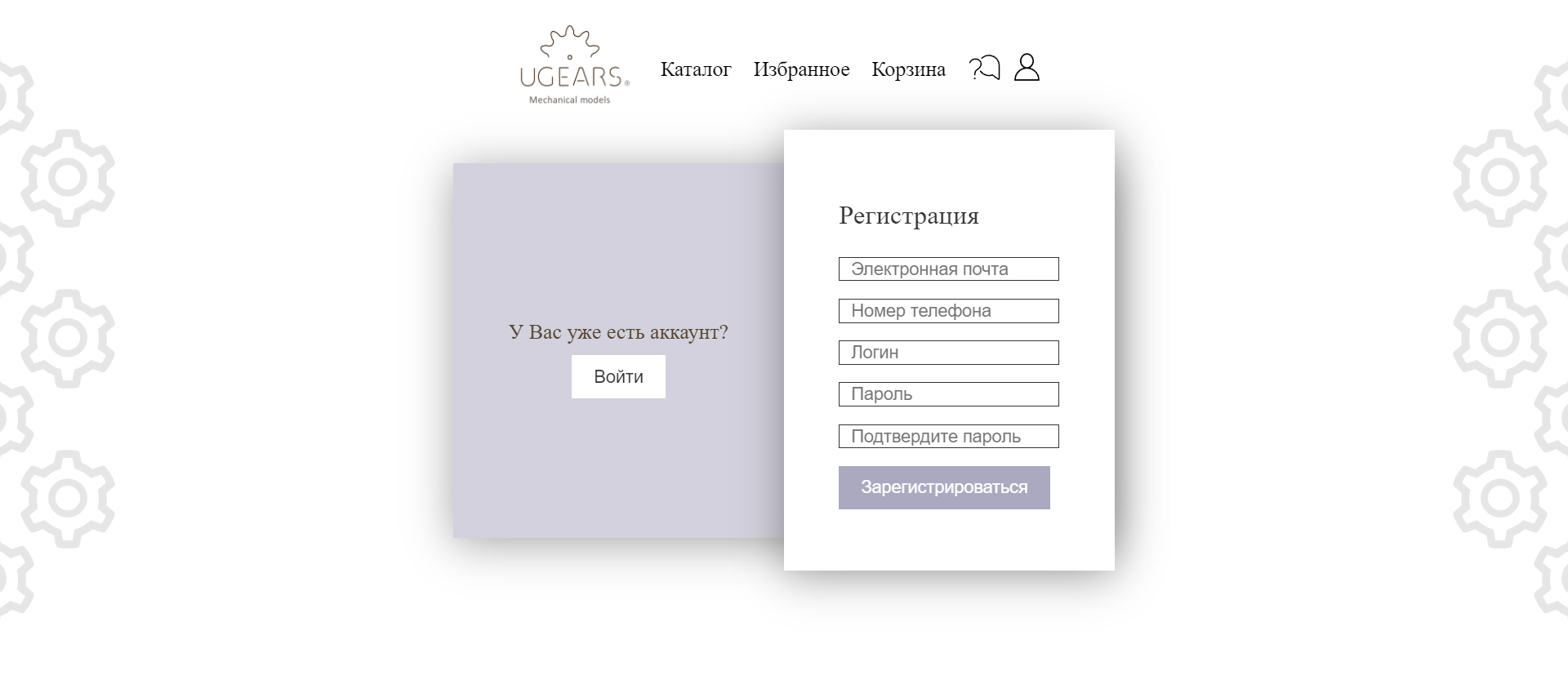


Рисунок 6. - Регистрация пользователя