**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ   
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: **«АВТОМАТИЗАЦИЯ СЛУЖБЫ УСЛУГ РЕМОНТНО-СТРОИТЕЛЬНОЙ КОМПАНИИ»**

Исполнитель: студент гр. ИП-21

Михайловский Е.В.

Руководитель: старший преподаватель

Шибеко В.Н.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2021

**СОДЕРЖАНИЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| Введение | | | 3 |
| 1. Анализ предметной области | | | 4 |
|  | 1.1 Обзор существующих методов решения | | 4 |
| 2. Предметная область | | | 6 |
|  | 2.1 Описание предметной области | | 6 |
|  | 2.2 Возможные сценарии | | 9 |
| 3.Структура программного обеспечения | | | 13 |
|  | 3.1 Обзор используемых средств | | 13 |
|  | 3.2 Структура базы данных | | 15 |
|  | 3.3 Классы слоя DataAccess | | 19 |
|  | 3.4 Классы слоя BusinessLogic  3.5 Классы слоя WebUI  3.6 Описание пользовательского графического интерфейса | | 21  23  26 |
| 4. Тестирование | | | 38 |
|  | | 4.1 Верификация | 38 |
| Заключение | | | 39 |
| Список использованных источников | | | 40 |
| Приложение А – Листинг программы | | | 41 |

**ВВЕДЕНИЕ**

В современном мире повсеместно внедряются технологии, на замену старому приходит новое и так происходит в каждой области нашей жизни. Совершенствуются технологии и постепенно проводится автоматизация любой работы.

Процесс автоматизации очень важен в современном мире, так как автоматизация позволяет тратить время с большей эффективностью. Так же, если автоматизация сделана правильно, то это позволяет выиграть всем, как компаниям, желающим автоматизировать свое производство или работу своих служб, так и клиентам, желающим в кротчайшие сроки выполнения своей задачи с наибольшей эффективностью.

Справедливо будет предположить, что для всего выше сказанного компаниям понадобятся различного рода приложения. Данные приложения должны будут обладать тем функционалом, который будет удобен в использовании и интуитивно понятен. Так же логичным будет предположить, что клиенту, использующему такое приложение, для осуществления заказа у компании необходимо видеть какой-либо справочником, который будет располагать видом предоставляемых услуг, также различные бригады осуществляющие какие-либо услуги и прейскурант. На внутреннем уровне приложение должно обладать функциями ведения справочников, осуществления заказа, поиска свободной бригады на заказ, оплаты и закрытия заказа. Также в приложении должна быть различного рода статистика, для улучшения процессов автоматизации.

Для создания данного приложения будет использоваться язык программирования C#, среда разработки *Visual Studio 2019,* возможности объектно-ориентированного программирования и работы с базами данных.

**АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

* 1. **Описание объекта**

Существует множество сервисов в той, или иной мере реализующих данные требования к данному курсовому проекту. Они охватывают, возможно, даже больший спектр услуг, чем, то приложение, которое разработано в данном проекте.

Первый ресурс, располагающий подобным функционалом – *kabanchik.by.* Кабанчик – это сервис поиска частных специалистов для решения бытовых, а также решения бизнес задач. Данный сервис предоставляет возможность клиенту сделать заказ на услуги вида: домашний мастер, отделочные работы, клининговые услуги, курьерские услуги, строительные работы, ремонт техники, логистические и складские услуги, бытовые услуги, мебельные работы и прочее. Потенциальный клиент оставляет заявку на нужный ему вид услуг и ждет, пока сервис подберет ему команду или специалиста для ее выполнения. В сервисе существует отчетность для выбора лучшего варианта. Обобщая все выше написанное, можно заявить, что данный сервис располагает полным спектром возможностей необходимых для комфортной работы как клиентам, так и компаниям, предоставляющим услуги.

Второй ресурс, реализующий нужды курсового проекта – *redsale.by*. Это бесплатный сервис по подбору частных специалистов. В нем существует возможность найти специалистов в сферах ремонта жилых помещений, ремонта техники, строительства, грузоперевозки, ремонта автомобилей. Данный сервис проводит автоматизацию работы компаний, которые им пользуются, что непосредственно влияет на их продуктивность. Автоматизация облегчает жизнь как клиентам, так и исполнителям, что не может не сказываться на положительном впечатлении от использования данного сервиса. На *redsale.by* существует свой прейскурант, ведение статистики и, прочий функционал, соответствующий решению поставленной задачи.

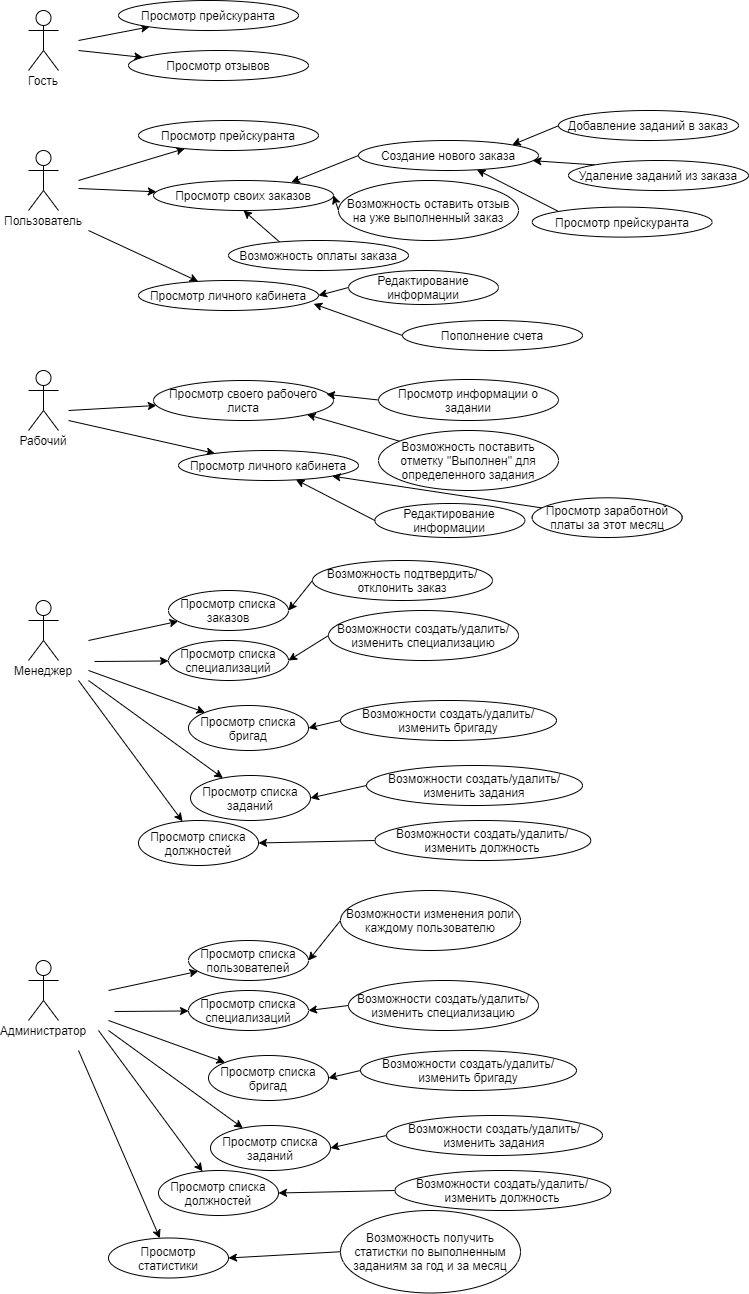
Третий ресурс строительная биржа *Remline.by* – это уникальная биржа в сфере строительства, которая позволяет создать тендер частному лицу, что позволяет сильно сэкономить денежные средства. *Remline* сервис обладающий всем необходимым функционалом для решения задач курсового проекта, в нем можно создать заказ, сервис его обработает и подберет свободную команду для его выполнения, для услуг существует прейскурант, так же существуют отзывы клиентов и примеры работ различных команд. Сервис проводит автоматизацию работ ремонтно-строительных служб, что позволяет им выполнять свою работу с наибольшей эффективностью.

Проанализировав приведенные ранее сервисы, можно сделать вывод о том, какой основной функционал должен быть у приложения, а именно: оформление, подтверждение и выполнение заказов, хранения базы данных о клиентах, их заказах, различного рода отчеты для администраторов приложений.

1. **ПРЕДМЕНТНАЯ ОБЛАСТЬ**
   1. **Описание предметной области**

Предметная область должна описывать все объекты, которые являются частью предметной области, а также любые взаимоотношения между ними. Следовательно, необходимо описать все зависимости между сущностями. Важно знать какие именно объекты попадают в предметную область и какие свзями между ними существуют и необходимо помнить цель проектирования данного приложения.

Для формирования представления о предметной области используют UML диаграммы, которые представлены на рисунке 1.1.

****

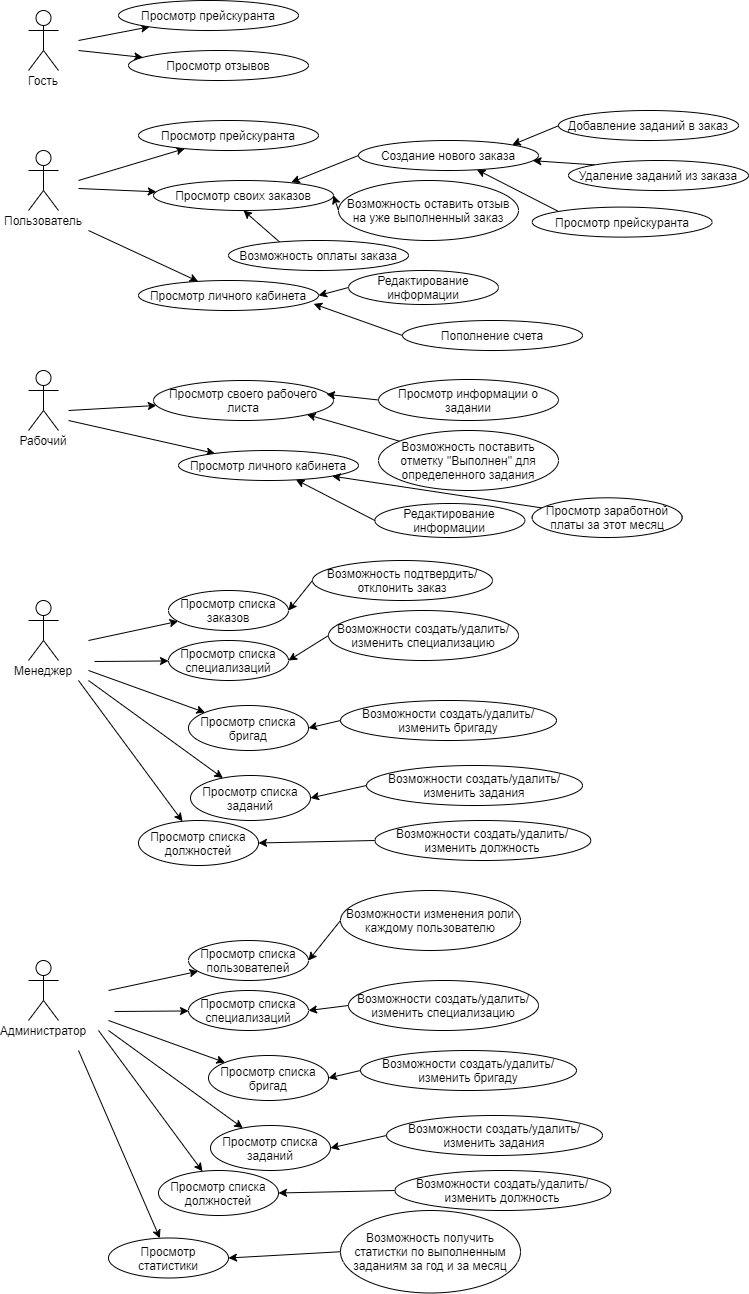
****

Рисунок 2.1 – Диаграмма прецендентов и актёров

Описание прецендентов и актеров.

* Актёр “Гость” – актер с минимальными возможностями, может просматривать прейскурант и отзывы.
* Актёр “Пользователь” – актер с возможностью создавать заказ и с возможностью его оплаты, так же может оставить откзыв на уже выполненный заказ. Как и “гость” имеет возможность просматривать прейскурант.
* Актёр “Менеджер” – актер, основаная функция которого – подтверждать или отклонять поступивший от пользователя заказ. Так же может редактировать списки бригад, заданий, специализаций, должностей.
* Актёр “Администратор” – актер, который обладает следующими полномочиями: управление списком пользователей, а именно изменением ролей, редактированием списков бригад, заданий, специализаций, должностей, так же получение статистики как за год, так и за выбранный месяц.
* Актер “Рабочий” – актер, обладающий функцией просмотра своего рабочего листа, в котором может отмечать выполненные задания, из количества выполненных заданий и их стоимости рассчитывается заработная плата для данного актера.

Описание прецедентов:

* Прецедент “просмотр отзывов” – прецедент для просмотра всех отзывов, оставленных пользователями, отображается по умолчанию на главной странице изображения.
* Прецедент “прейскурант” – просмотр всех заданий, которые может выполнить данная компания.
* Прецедент “добавление задания” – создание нового задания по указанной специализации.
* Прецедент “изменение задания” - изменение указанного задания.
* Прецедент “удаление задания” – прецедент для удаления услуги.
* Прецедент “контроль пользователей” – прецедент для просмотра учетных записей всех пользователей и, при необходимости изменения им ролей.
* Прецедент “регистрация” – возможность зарегистрироваться новому пользователю.
* Прецедент “редактирование учётной записи” - изменение учётной записи, осуществляется самим пользователем этой учетной записи
* Прецедент “просмотр заказов” – прецедент вызываемый или от пользователя, или от менеджера, пользователь имеет возможность создать новый заказ и удалить его, до тех пор, пока не была произведена оплата. Менеджер имеет возможность просмотреть все заказы и подтвердить или же отклонить оплаченные заказы.
* Прецедент “подтверждение готовности задания” – прецедент вызываемый работничим в случае, если он выполнил задание, заказ считается закрытым, когда все задания из него выполнены.
* Прецедент “просмотр статистики” – прецедент, предназначенный для администратора, в котором он может просмотреть отчетность бригад по количеству выполненных заданий.
* Прецедент “прейскурант” – возможность просмотреть список предоставляемых услуг и диапазон цен на нее.
* Прецедент “написать отзыв” – после выполнения всех заданий из заказа пользователь может оставить отзыв, который будет виден на главной странице приложения.
* Прецедент “управление заказами” – клиент может создать новый заказ. После создания заказа, у пользователя появляется возможность добавить в него задания и удалять их, если существует потребность. После добавления всех заданий в заказ, пользователь может его оплатить.
* Прецедент “добавление специализации” – создание новой специализации под нужны компании.
* Прецедент “изменение специализации” – изменение существующей специализации.
* Предент “удаление специализации” – удаление существующей специализации.
* Прецедент “добавление бригады” – создание новой бригады под нужны компании.
* Прецедент “изменение бригады” – изменение существующей бригады.
* Предент “удаление бригады” – удаление существующей бригады.
* Прецедент “добавление должности” – создание новой должности под нужны компании.
* Прецедент “изменение должности” – изменение существующей должности.
* Предент “удаление должности” – удаление существующей должности.
  1. **Возможные сценарии**

Сценарий прецедента “создание заказа”:

1. Отображение всех заказов пользователя.
2. Пользователь нажимает на кнопку создания нового заказа.
3. Указываются название и описание заказа.
4. Подтверждение заказа.

Сценарий прецедента “управление заказами”:

1. Отображение всех заказов пользователя.
2. Пользователь указывает в какой заказ хочет добавить задание.
3. Отображение подробностей о заказе.
4. Пользователь нажимает на кнопку добавления задания.
5. Выбор специализации задания.
6. Выбор соответствующего специализации задания.
7. Выбор свободной даты для заказа.
8. Написание краткого описания для заказа, например адресс проведения работ.
9. Заказ добавлен.

Сценарий прецедента “подтверждение оплаченного заказа”:

1. Отображение всех заказов.
2. Менеджер выбирает нужный оплаченный заказ.
3. Менеджер выбирает или подтвердить заказа, или отклонить, если он не соответствует нормам.
4. Заказ подтвержден.

Сценарий прецедента “изменение роли пользователя”:

1. Отображение.
2. Рекрутер нажимает на кнопку “изменить учётную запись”.
3. Система открывает диалоговое окно для ввода номера учётной записи для изменения.
4. Рекрутер или вводит номер учётной записи и нажимает на кнопку “изменить” или нажимает на кнопку “отмена” (если “отмена”, то вариант использования завершается).
5. Если рекрутер нажал изменить, то открывается следующее диалоговое окно, в котором он изменяет поля выбранной учётной записи.
6. Если рекрутер далее нажимает “сохранить изменения”, то учётная запись перезаписывается, если же отмена, то вариант использования завершается.
7. Вариант использования завершается.

Сценарий прецедента “регистрация”:

1. Пользователь, находясь на главной странице, нажимает на кнопку зарегистрироваться.
2. Отображается страница с формой регистрации, где пользователь должен указать имя, фамилию, свою почту, ввести и подтвердить пароль.
3. Если пароль прошел проверку на надежность и пользователя с такой же почтой не существует, то регистрация прошла успешно.

Сценарий прецедента “прейскурант”:

1. Пользователь, находясь на главной странице нажимает на кнопку прейскурант.
2. Приложение отображает все возможные задания, которые могут выполнять бригады компании.
3. В случае если несколько бригад могут выполнять одно и то же задание с разной стоимостью, тогда приложение отображает диапазон возможных цен.

Сценарий прецедента “добавление специализации”:

1. Менеджер или администратор, находясь в главном меню, нажимают на кнопку “специализации”.
2. Отображаются все возможные специализации.
3. Менеджер или администратор нажимают на кнопку “создать новую специализацию”.
4. Отркывается окно с готовой формой бригады, где необходимо указать название специализации и ее описание.
5. При нажатии на кнопку подтверждение специализация добавляется в список специализаций.

Сценарий прецедента “добавление отзыва”:

1. Пользователь, находясь на главной странице переходит на страницу с его заказами.
2. В случае если заказ в состоянии “выполнен”, тогда у пользователя появляется возможность оставить отзыв.
3. После нажатия на кнопку “отзыв”, появляется форма, в которую пользователь может ввести свой отзыв на заказ.
4. В случае если пользователь ввел отзыв – он появляется на главной странице.

Сценарий прецедента “изменение бригады”:

1. Менеджер или администратор, находясь на главной странице выбирают кнопку “Бригады”.
2. Приложение отображает список всех бригад и выбирается та бригада, которую необходимо изменить.
3. После нажатия на кнопку “изменить”, появляется форма, которую необходимо заполнить новой информации по этой бригаде.
4. После нажатия на кнопку “изменить”, на странице бригад, информация по выбранной бригаде уже будет обновлена.

Сценарий прецедента “удаление должности”:

1. Менеджер или администратор, находясь на главной странице, нажимают на кнопку “должности”.
2. Появляется список всех должностей, где можно выбрать должность, которую необходимо удалить.
3. После того, как была нажата кнопка “удалить”, появляется форма, на которой отображается вся информация об удаляемой должноти, и кнопка “подтвердить”.
4. После подтвеждения удаления, выбранная бригада уже не отображается в списке всех бригад.

Сценарий прецедента “просмотр статистики”:

1. Администратор, находясь на главной странице, нажимает на кнопку “статистика”.
2. Приложение отображает форму, на которой можно выбрать статистика за какой период необходима.
3. При нажатии на кнопку “статистика за год”, появляется график, где отображаются все бригады и все задания, которые они выполнили за год.
4. При нажатии на кнопку “статистика за месяц”, появляется форма, где необходимо выбрать за какой месяц неоходима статистика, после выбра месяца появляется график с бригадами и заданиями, которые они выполнили за выбранный месяц.

**3 СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

* 1. **Обзор используемых средств**

Чтобы создать необходимую программу был выбран язык программирования C#, на платформе Microsoft .NET Framework и основной технологией ASP.NET. Данная технология обладает всеми необходимыми возможностями для создания необходимого приложения.

ASP.NET — бесплатная интернет-платформа для создания веб-сайтов и веб-приложений с помощью HTML, CSS, JavaScript, C#. Также можно создавать веб-API и использовать технологии реального времени, такие как веб-сокеты.

ASP.NET предоставляет три платформы для создания веб-приложений: веб-формы, ASP.NET MVC и веб-страницы ASP.NET. Все три платформы стабильны и полноценны: веб-приложения можно создать с помощью любой из них. Независимо от того, какая платформа будет выбрана, везде будет возможность получить все преимущества и возможности ASP.NET.

Каждая платформа предназначена для определенного стиля разработки. Выбор зависит от сочетания навыков программирования (знаний, опыта разработки), типа создаваемого приложения и удобного подхода к разработке.

Из-за особенностей данного проекта, была выбрана платформа веб-страницы ASP.NET. Веб-страницы ASP.NET и синтаксис Razor обеспечивают быстрый, понятный и простой способ объединения серверного кода с HTML для создания динамического веб-содержимого.

Для хранения данных используется база данных MS SQL. Для удобного просмотра данных было использовано средство администрирования SQL Server Management Studio.

Microsoft SQL Server — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL" \o "Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase" \o "Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, появляется возможность ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Для хранения, обработки и использования информации о пользователях был выбран ASP.NET Identity

ASP.NET Identity представляет встроенную в ASP.NET систему аутентификации и авторизации. Данная система позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями или использовать для входа на сайт учетные записи внешних провайдеров, таких как Facebook, Google, Microsoft, Twitter и других.

* 1. **Структура базы данных**

****

Рисунок 3.1 – Стуктура базы данных

Описание таблиц базы данных:

Таблица “Order” предназначена для хранения основной информации о заказе. Поле “Id” хранит номер записи в таблице, поле “Title” хранит название, поле “IdCustomer” хранит номер пользователя, создавшего заказ, поле “OrderStatus” хранит состояние заказа на текущий момент, поле “Requirements” хранит описание заказа.

Таблица “Feedback” хранит информацию об отзывах на заказы. Поле “Id” хранит номер записи в таблице, поле “IdOrder” хранит номер заказа, который связан с записью, поле “Review” хранит текст отзыва.

Таблица “Customer” хранит основную информацию о пользователе. Поле “Id” хранит номер пользователя, поле “IdentityUser” хранит номер записи о пользователе в таблице “AspNetUsers”.

Таблица “AspNetUsers” хранит информацию о пользователях. Поле “Id” хранит номер пользователя в таблице, поля “FirstName”, “Surname”, “LastName” хранят фамилию имя и отчетсво пользователя, поле “Balance” хранит балан пользователя, поля “Email” и “EmailConfirmed” хранят электронную почту пользователя и прошла ли она проверку, поля “PhoneNumber” и “PhoneNumberConfirmed” хранят телефонный номер пользователя и прошле ли он проверку, поле “UserName” хранит имя пользователя, которое может отображаться в его аккаунте.

Таблица “AspNetLogins” отвечает за возможность входа в систему. Поле “LoginProvider” хранит логин входящего, поле “ProviderKey” хранит ключ входящего, поле “UserId” хранит номер пользователя.

Таблица “AspNetRoles” хранит информацию о возможных ролях. Поле “Id” хранит уникальный номер роли, поле “Name” хранит название роли.

Таблица “AspNetUserRoles” хранит информацию о том, какая у пользователя роль. Поле “UserId” хранит номер пользователя, поле “RoleId” хранит номер роли.

Таблица “AspNetClaims” хранит требования к пользователям. Поле “Id” хранит номер записи в таблце, поле “UserId” хранит номер пользователя, поле “ClaimType” хранит тип требования, “ClaimValue” хранит значение требования.

Таблица “OrderTask” является ассоциотивной между таблицами “Order” и “Task”. Поле “Id” хранит номер записи в таблице, поле “OrderId” хранит номер заказа, поле “TaskId” хранит номер задания, поле “TaskCompletionDate” хранит дату выполнения заказа, поле “Status” хранит статус задания, поле “Description” хранит описание задания.

Таблица “Task” хранит информацию о заданиях. Поле “Id” хранит номер записи, поле “Title” хранит название задания, поле “IdSpecialization” хранит номер специальности для данного задания, поле “Price” хранит стоимость задания, поле “Description” хранит описание задания, поле “IdBrigade” хранит номер бригады, которая будет выполнять это задание.

Таблица “Brigade” хранит информацию о бригадах. Поле “Id” хранит номер записи, поле “Title” хранит название бригады, поле “IdSpecialization” хранит номер специализации этой бригады.

Таблица “Specialization” хранит информацию о специализациях. Поле “Id” хранит номер записи, поле “Name” хранит название специализации, поле “Description” хранит описание специальности.

Таблица “Employee” хранит информацию о работниках. Поле “Id” хранит номер записи в таблице, поле “IdBrigade” хранит номер бригады, к которой отностится рабочий, поле “Selary” хранит коэффициент заработной платы, поле “IdJobPosition” хранит номер должности, поле “IdentityUserId” хранит номер пользователя в таблице “AspNetUsers”.

Таблица “JobPosition” хранит данные о должностях. Поле “Id” хранит номер записи, поле “Title” хранит название должности, поле “Purpose” хранит описание должности.

Таблица “Manager” хранит основную информацию о менеджерах. Поле “Id” хранит номер записи, поле “Selary” хранит коэффициент заработной платы, поле “IdentityUserId” хранит номер пользователя в таблице “AspNetUsers”.

Таблица “\_MigrationHistory” хранит историю входов в приложение. Поле “MigrationId” номер записи в таблице, поле “ContextKey” хранит ключ контекста, поле “Model” хранит модель входа, поле “ProductVersion” хранит версию приложения.

* 1. **Классы слоя DataAccess**

Классы слоя DataAccess предназначены для получения данных из базы данных.

Классы Entities полностью повторяют таблицы из базы данных и служат для хранения информации из таблиц на слое DataAccess.

В качестве примера описательного класса возьмем класс Manager.

Класс Manager служит для хранения информации о менеджерах из таблицы “Managers” содержит такие поля как:

* int Id – поле хранит номер менеджера из соответствующей таблицы.
* string Salary – поле хранит коэффициент заработной платы.
* string IdentityUserId – поле хранит строковый номер менеджера из таблицы AspNetUsers.

Классы Repositories предназначены для обращения к базе данных от соответствующих классов Enities, реализуют интерфейс IRepositories<T>. Реализует такие методы как, на примере ManagerRepositories:

* public int Create(Manager item) – метод для создания новой записи в таблице базы данных.
* public void Delete(int id) – метод для удаления записи из таблицы базы данных.
* public IEnumerable<Manager> GetAll() – метод получения всех записей из таблицы базы данных.
* public Manager GetById(int id) – метод получения записи по ее номеру из таблицы базы данных.
* public void Update(Manager item) – метод для обновления информации о записи в таблице из базы данных.

Интерфейс IRepositories<T> описывает CRUD модель с дополнительным методом GetById() для всех репозиториев.

Классы Enums предназначены для хранения данных типа enum, в данном случае хранят два класса OrderStatus и OrderTaskStatus, которые в свою очередь содержат состояния заданий и заказов.

* 1. **Классы слоя BusinessLogic**

Классы этого слоя необходимы для обработки данных полученых со стоя DataAccess и последующей передачи информации на слой WebUI. Так, нагрузка на приложение будет разделена на серверную и пользовательскую часть, где на пользовательской части будут создаваться запросы, которые будут отправляться на сервер, где они и будут обрабатываться.

Классы Dtos повторяют классы Enities на слое DataAccess.

Классы Interfaces реализуют интерфейсы для существующих сервисов, в данном случае это сервисы BrigadeService, OrderService, ReportService, UserService, реализующие методы необходимые для выполнения поставленных задач.

Рассмотрим BrigadeService из Services, он отвечает за методы, неободимые для управления всего, что связано с бригадами. В этом классе были созданы репозитории специализаций, бригад, заданий, заказ-заданий, должностей, рабочих, для обращения к соответствующим репозиториям со слоя DataAccess. Сервис реализует такие методы как:

* public int CreateEmployee(EmployeeDto item) – обращается к репозиторию работника для создания новой записи в таблице “Employee”.
* public void DeleteEmployee(int id) – обращается к репозиторию работника для удаления записи из таблицы “Employee”.
* public IReadOnlyCollection<EmployeeDto> GetAllEmployees() – обращается к репозиторию работника для получения всех записей из таблицы.
* public EmployeeDto GetEmployeeById(int id) – обращается к репозиторию работника для получения записи по номеру.
* public void UpdateEmployee(EmployeeDto item) – обращается к репозиторию работника для обновления информации в таблице.
* public void ValidateEmployee(EmployeeDto item) – метод предназначенный для валидации вышеописанных методов.

Такие же методы реализуются для специализаций, должностей, бригад, заказ-заданий, заданий.

* public List<TaskDto> FindTasksBySpecialization(int specializationId) – метод предназначенный для получения коллекции заданий по номеру их специализации.

BrigadeService, OrderService реализуют аналогичные методы для своих репозиториев, а так дополнительные методы такие как:

* public string GetCustomerIdentiryByOrder(int IdOrder) – получение IdentityUserId пользователя по номеру заказа.
* public int FindOrderTaskByOrderAndTaskIds(int orderId, int taskId) – поиск номер заказ-задания по номерам заказа и задания.
* public BrigadeDto FindFreeBrigadeForDate(DateTimeOffset date, int idSpecialization) – поиск свободной бригады на выбранную дату.
* public decimal GetSelaryByBrigadeId(int brigadeId) – получение зарплаты работника за текущий месяц по номеру его бригады.
* public void RemoveFromRoles(string id) – удаление роли пользователя по его IdentityUserId.

ReportService – сервис предназначенный для обработки запросов связанных с статистикой. Реализует такие методы как:

* public IList<ReportYearDto> GetReportForYear() – получение коллекции отчетов на каждый день за год.
* public IList<ReportYearDto> GetReportForMonth(int month) – получение коллекции отчетов на каждый день за выбранный месяц.
* public IList<MonthDto> GetAllMonth() – получение коллекции месяцов с соотвествующими номерами.

Класс BusinessLogicMapperProfile предназначен для проецирования классов Dtos на Entities из DataAccess.

Класс Constants предназначен для хранения различного рода константных значений для данного слоя.

* 1. **Классы слоя WebUI**

Классы слоя WebUI являются классами слоя представления, тут располагаются модели для страниц, представления страниц, контроллеры, которые получают и информацию со стоя BussinessLogic и представляют ее к виду, нужному для представлений, различного рода скрипты для отображения на странице.

Классы Models предназначенны для создания и хранения моделей, которые отображаются пользователю.

Описание класса EmployeeFullInfoViewModel из Models, являющийся потомком класса IdentityUser:

* public string Surname { get; set; } – свойтсво предназначенное для хранения фамалии работника.
* public string FirstName { get; set; } – свойство предназначенное для хранения имени работника.
* public string LastName { get; set; } – свойство предназначенное для хранения отчества работника.
* public decimal Balance { get; set; } – свойство предназначенное для хранения баланса работника.
* public string BrigadeName { get; set; } – свойство предназначенное для хранение названия бригады работника.
* public string JobPositionName { get; set; } – свойство предназначенное для хранения должности работника.

Подобным образом выглядят все классы из Models, они включают в себя различные поля, необходимые для отображения желаемой страницы.

Классы Controllers необходимы для заполнения моделей из Models и последующей передачи их на отображение на сайте.

Пример описания класса UsersController из Controllers:

* [HttpGet] public ActionResult Index() – метод получения информации для страницы Index. Возвращает коллекцию UsersIndexViewModel, которая будет отображаться на сайте.
* [HttpGet] public ActionResult Update(int id) – метод получения информации для страницы Update, получает номер пользователя информацию о котором необходимо обновить. Возвращает UsersIndexViewModel, которая будет отображаться на сайте.
* [HttpGet] public ActionResult ChangeRole(string id, string roleName) – метод изменения роли для выбранного пользователя. Производит поиск необходимого пользователя в базе данных, после чего меняет ему роль на заданную. Если выбранная роль “Менеджер” – вызывает страницу “SetSelary”, если выбранная роль “Рабочий” – вызывает страницу “ChangeSpecialization”, в остальных случаях меняет роль и вызывает “Index”.
* [HttpGet] public ActionResult ChangeEmployee(string id, int specId) – метод, который получает информацию для ChangeEmployeeViewModel, которую передает на страницу ChangeEmployee.
* [HttpGet] public ActionResult SetSelary(string id) – метод получающий информацию для страницы SetSelary.
* [HttpPost] public ActionResult SetSelary(SetSelaryViewModel model) – метод создающий нового менеджера исходя из информации полученной из прошлых страниц и возвращает пользователя на “Index”.
* [HttpPost] public ActionResult ChangeEmployee(ChangeEmployeeViewModel model) – метод удаляющий пользователя из его прошлой роли и таблицы базы данных и создающий ему роль работника и заносящий его в базу данных.
* [HttpGet] public ActionResult ChangeSpecialization(string id) – метод создающий модель UserChangeSpecializationViewModel и передающий ее на страницу ChangeSpecialization.

Классы Identity предназначены для работы с авторизацией, регистрацией и контроля пользователей, включают в себя: User – класс с информацией о пользователе и методе, который генерирует User Identity, IdentityConstants – класс, хранящий константы, ApplicationUserManager – класс проводящий валидацию пользователя, ApplicationSignInManager – класс создающий пользователя.

Класс WebUIMapperProfile аналогичен BussinesLogicMapperProfile, проводит проецирование классов Dtos на некоторые классы Models.

Классы Enums аналогичны таким же на слое DataAccess.

Классы Views хранят cshtml файлы с разметкой и выводом информации для каждой страницы.

* 1. **Описание пользовательского графического интерфейса**

Все современные приложения стараются делать интуитивно понятными и простыми в использованни, исходя из этого стоит уделить время для реализации хорошего графического интерфейса.

Так для незарегистрированного пользователя доступна лишь главная страница и страница с прейскурантом.



Рисунок 3.2 – Главная страница незарегистрированного пользователя

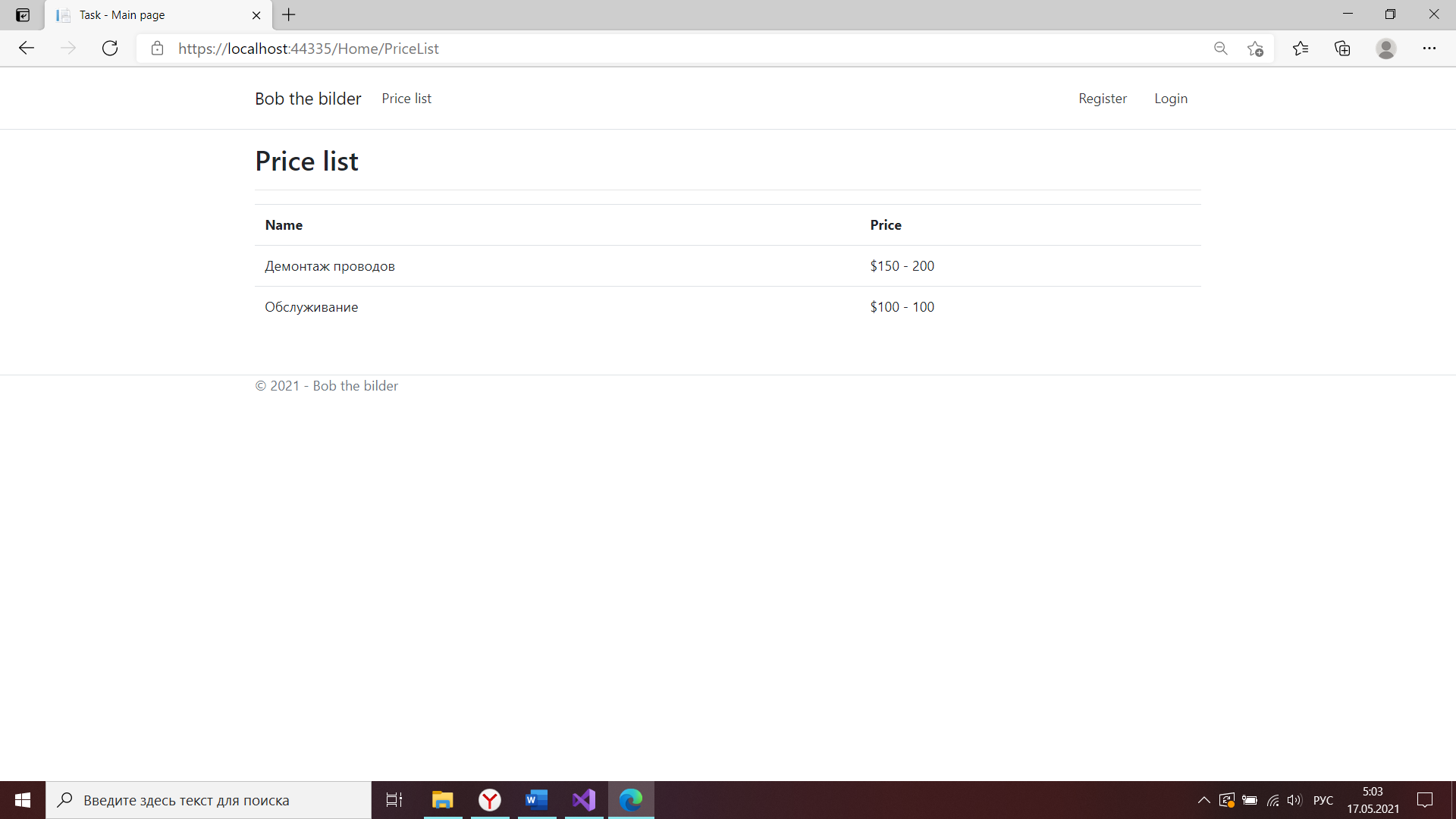


Рис 3.3 – Страница прейскуранта для всех пользователей

На главной странице у гостя есть возможность зарегистрироваться, если он новый пользователь или же зайти в свой аккаунт, если он регистрировался ранее.

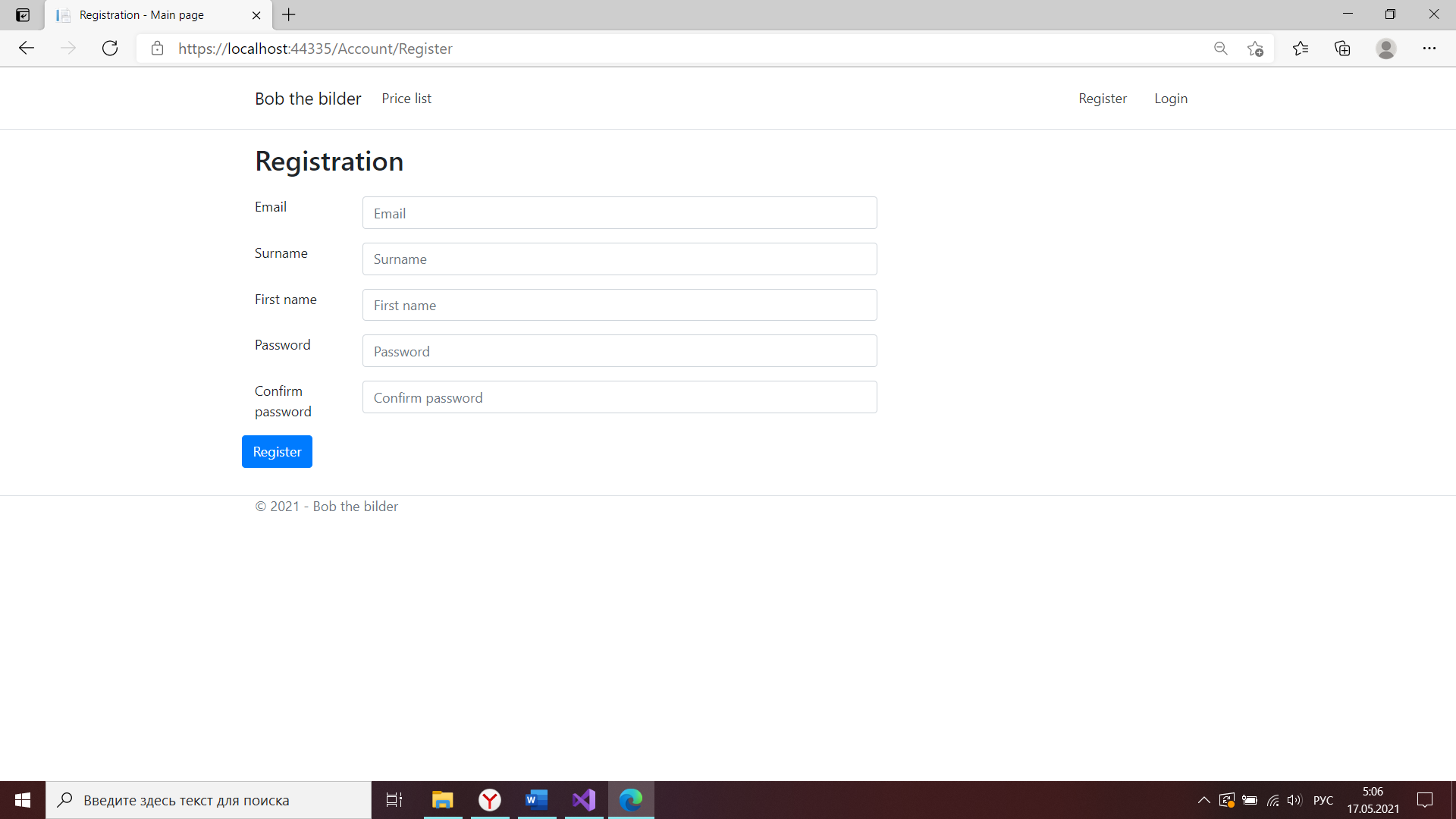


Рис 3.4 – Страница регистрации

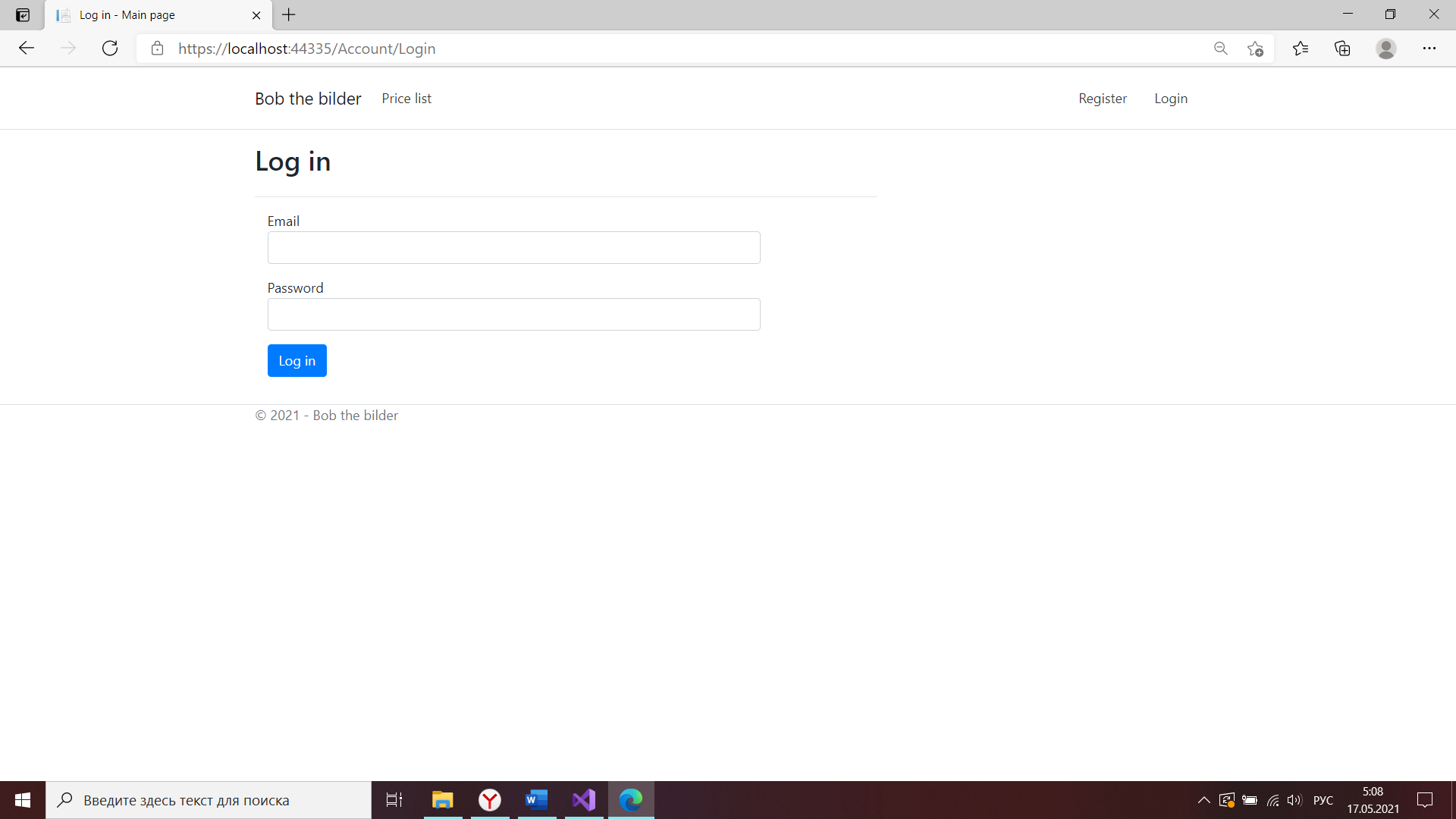


Рис 3.5 – Страница авторизации

После того, как пользователь вошел в систему его главная страница изменится.

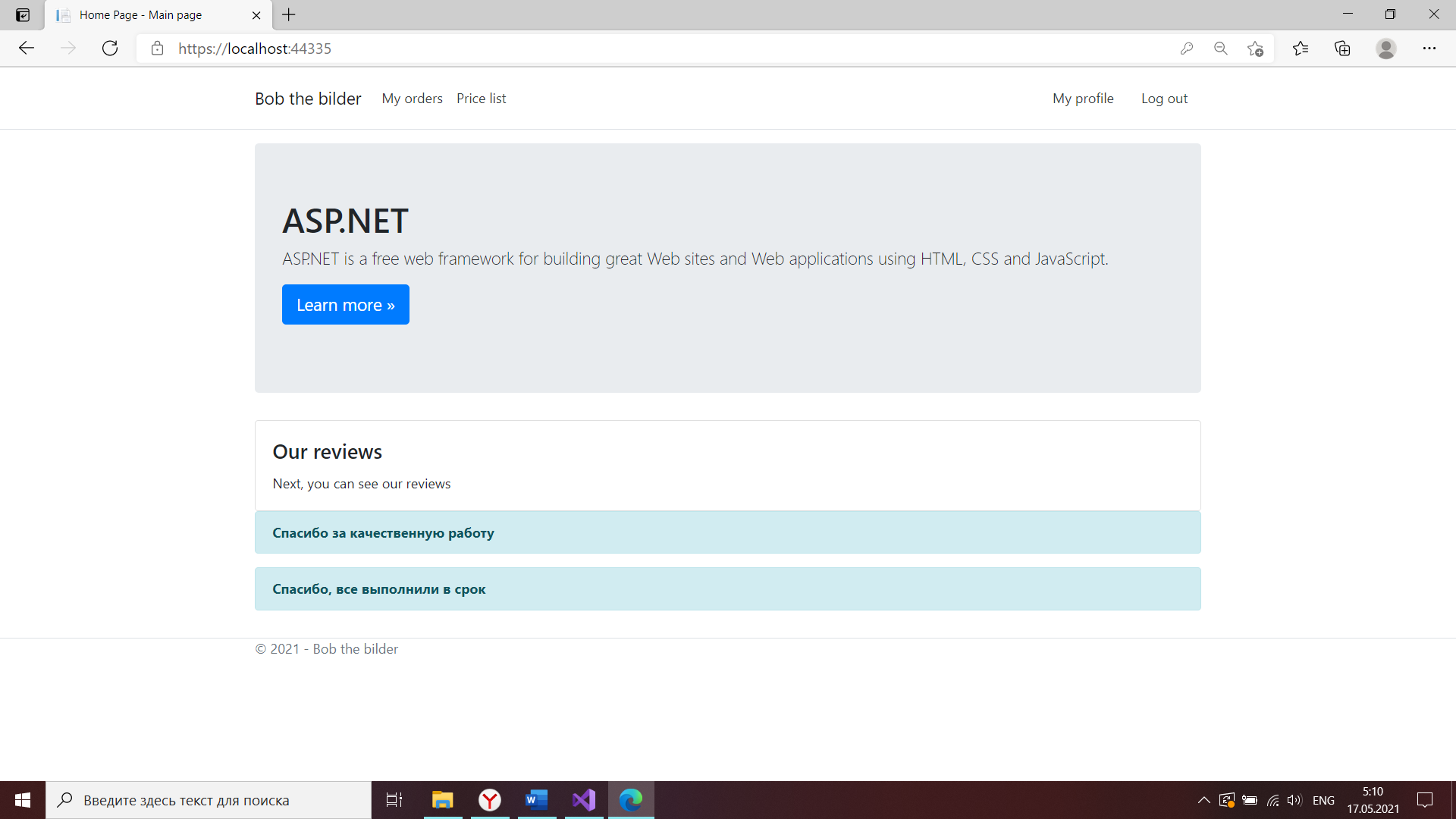


Рис 3.6 – Главная страница пользователя

Здесь пользователь может просмотреть свои заказы нажав на кнопку “My orders”.

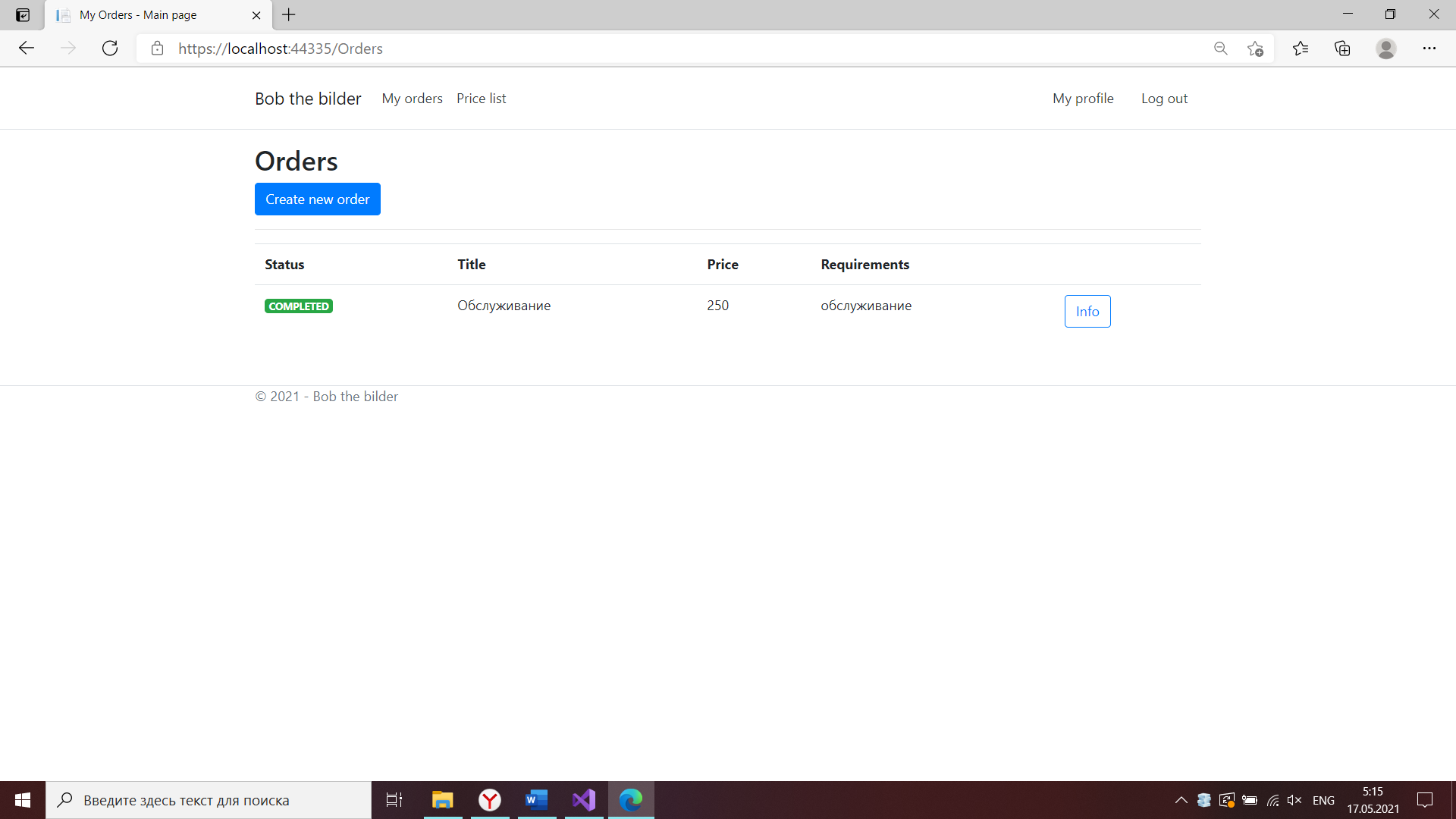


Рис 3.7 – Страница заказов пользователя

На этой странице пользователь имеет возможность просмотреть информацию о своих заказах, а также создать новый.



Рис 3.8 – Страница создания заказа

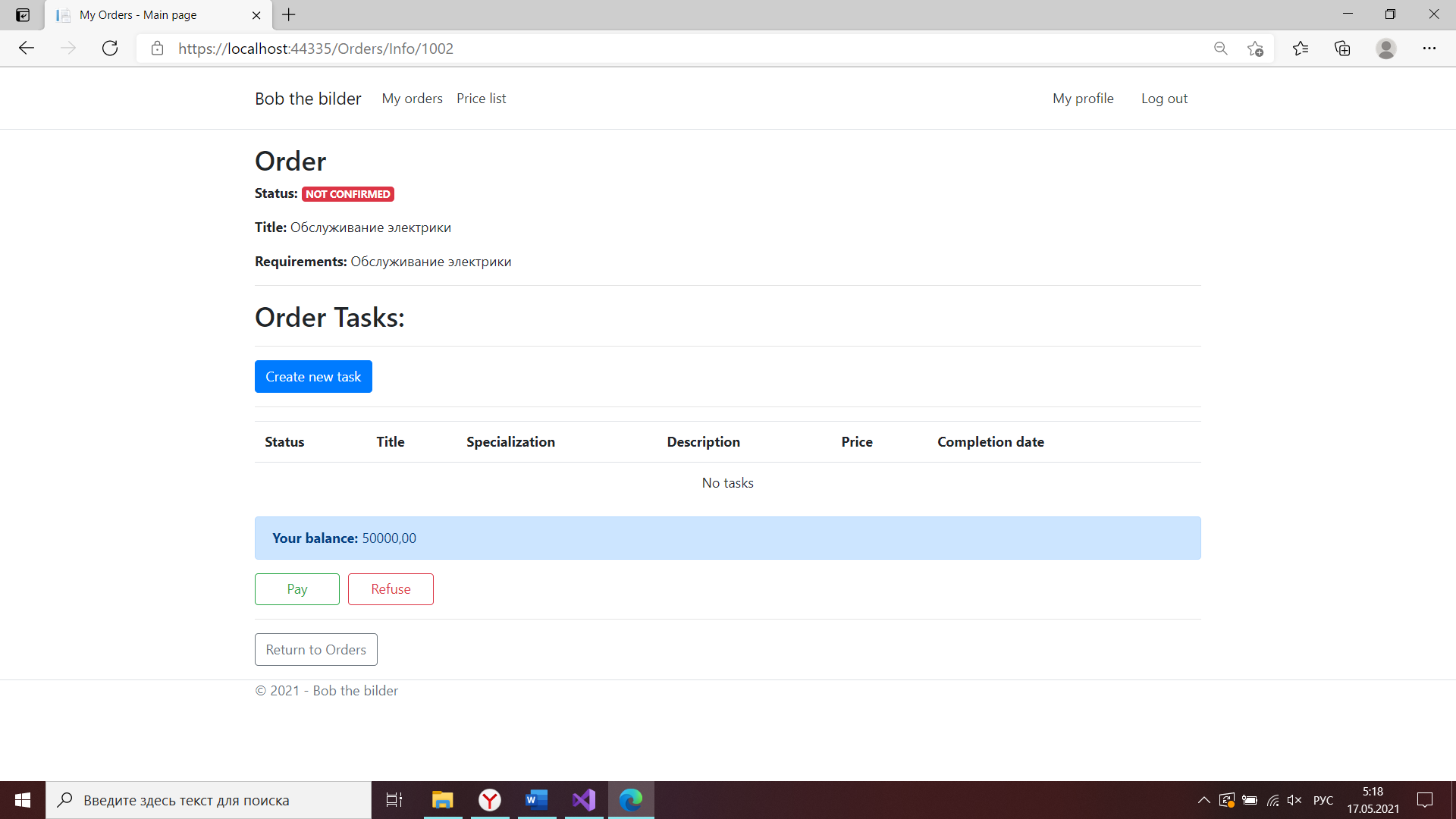


Рис 3.9 – Страница с информацией о заказе

На данной странице у пользователя есть возможность добавить задания в заказ, а также оплатить или отказаться от него.

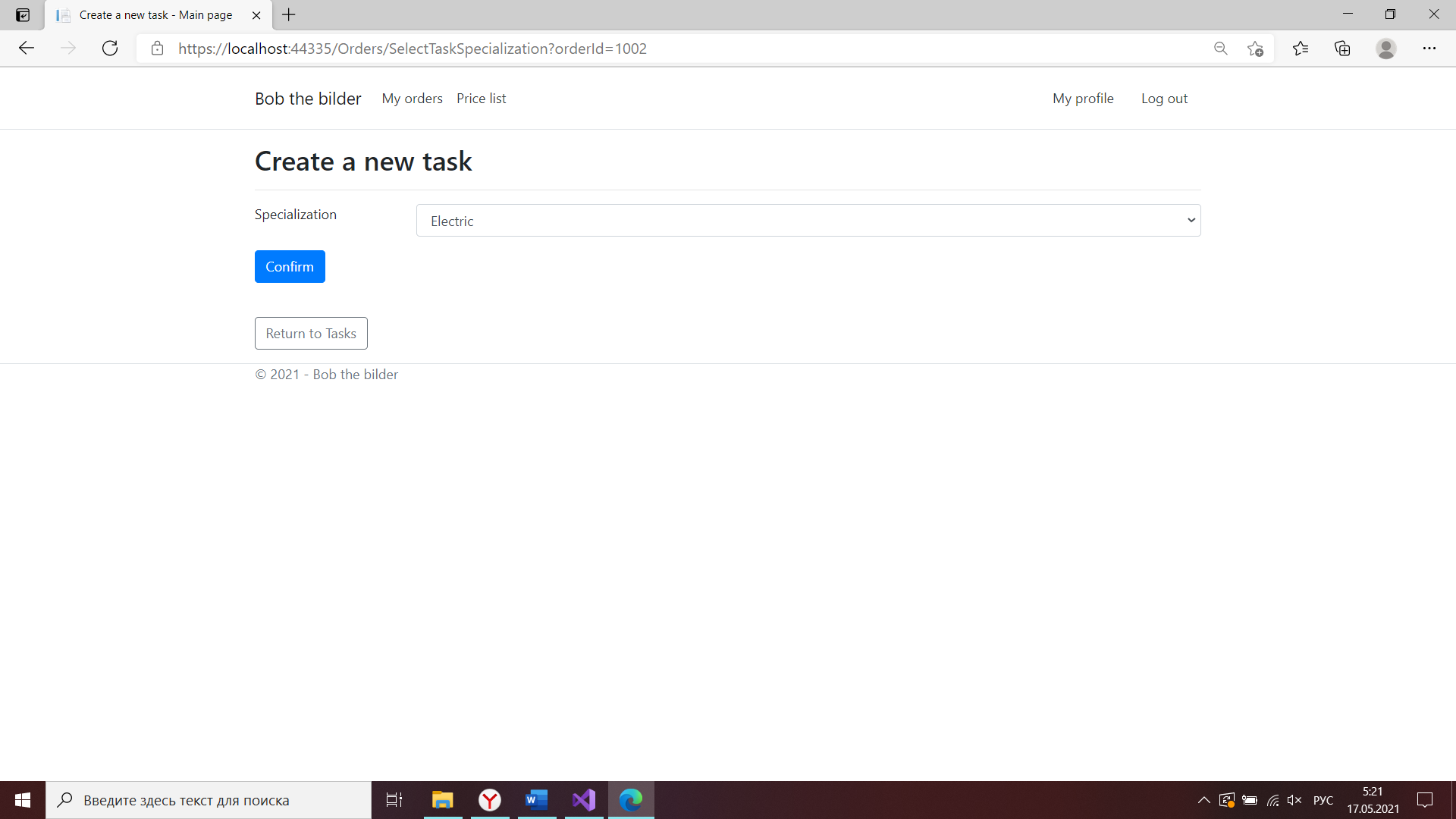


Рис 3.10 – Страница выбора специализации для заказа

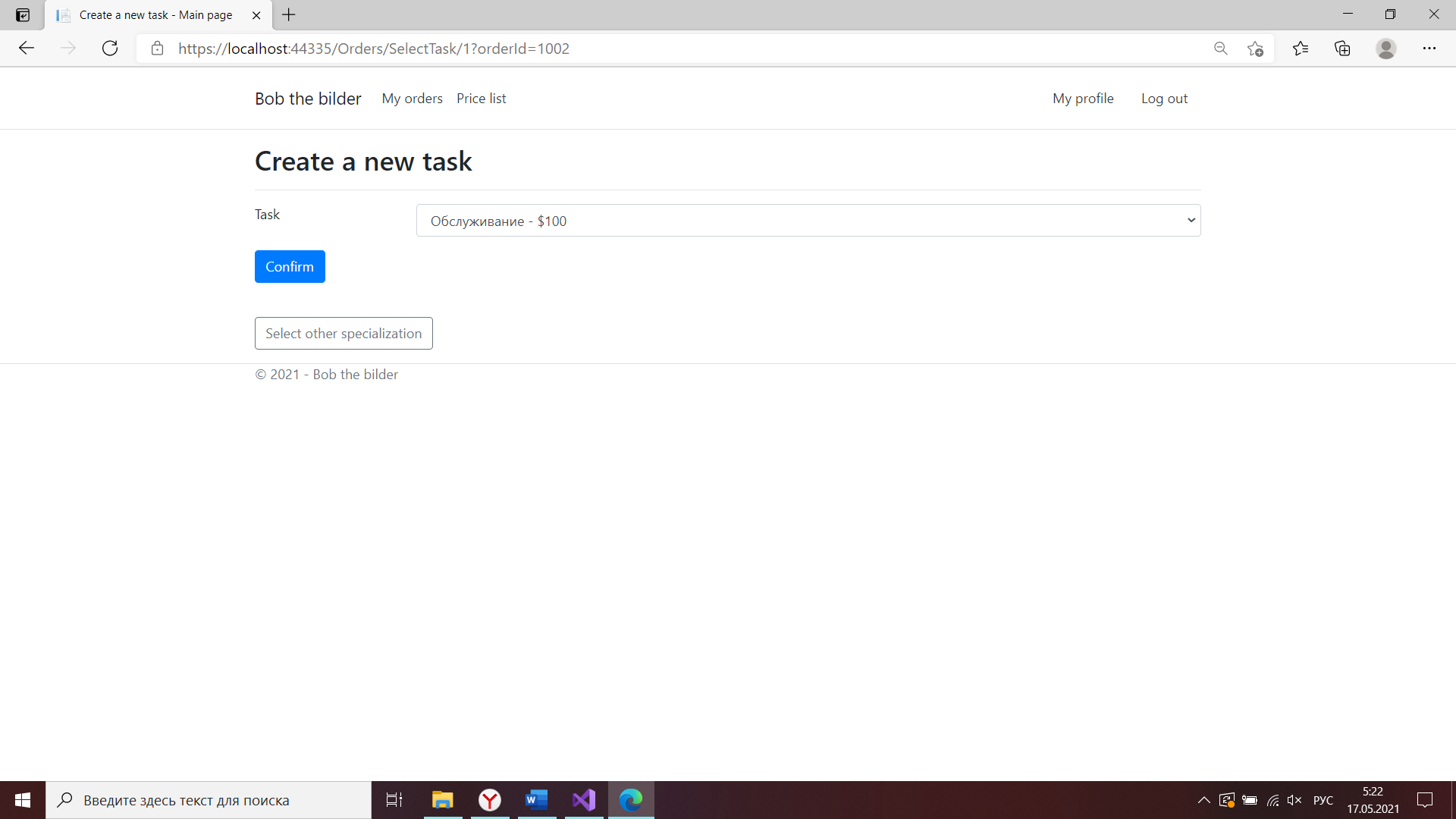


Рис 3.11 – Страница выбора задания для заказа

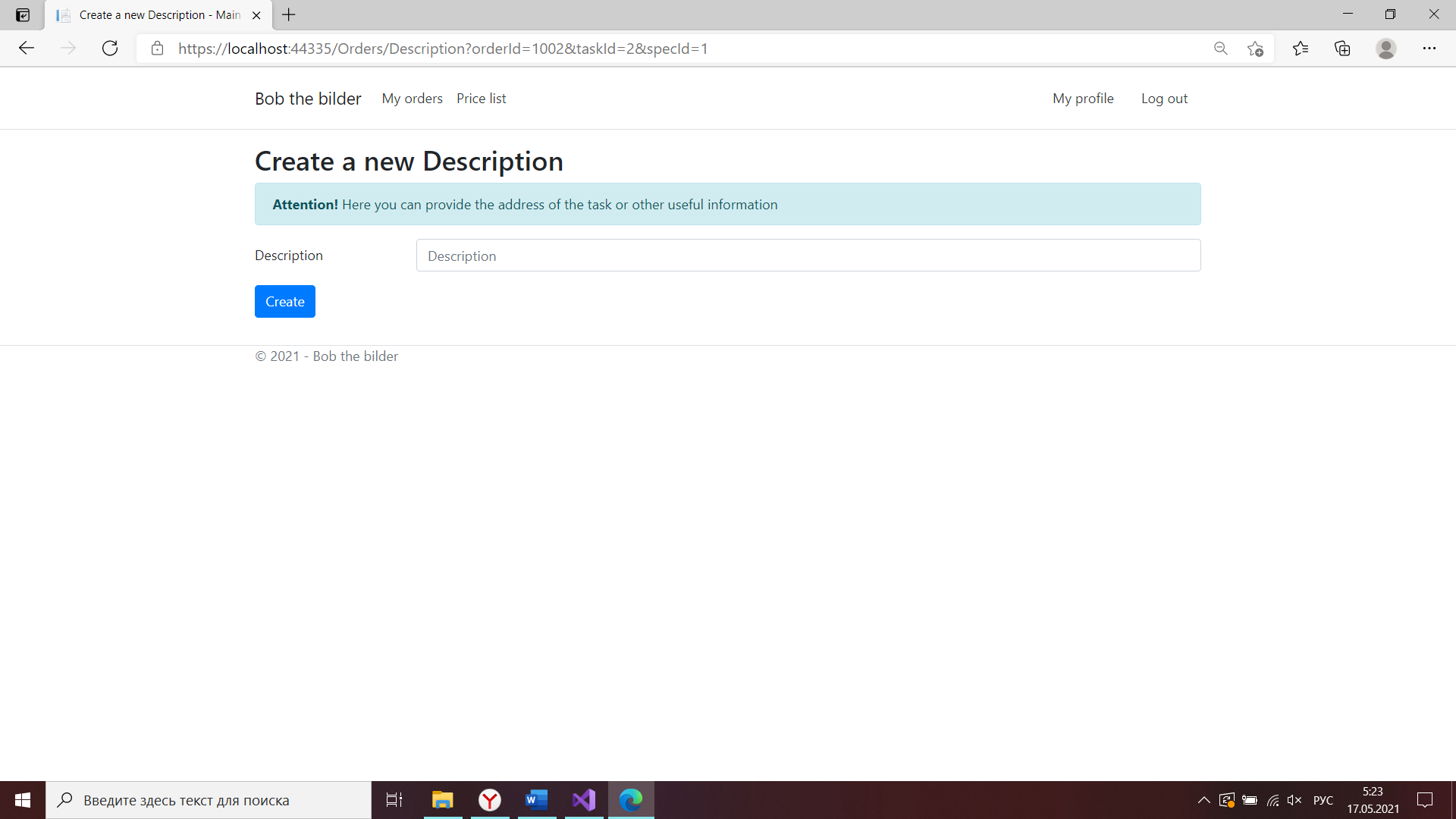


Рис 3.12 – Страница добавление описания для заказа

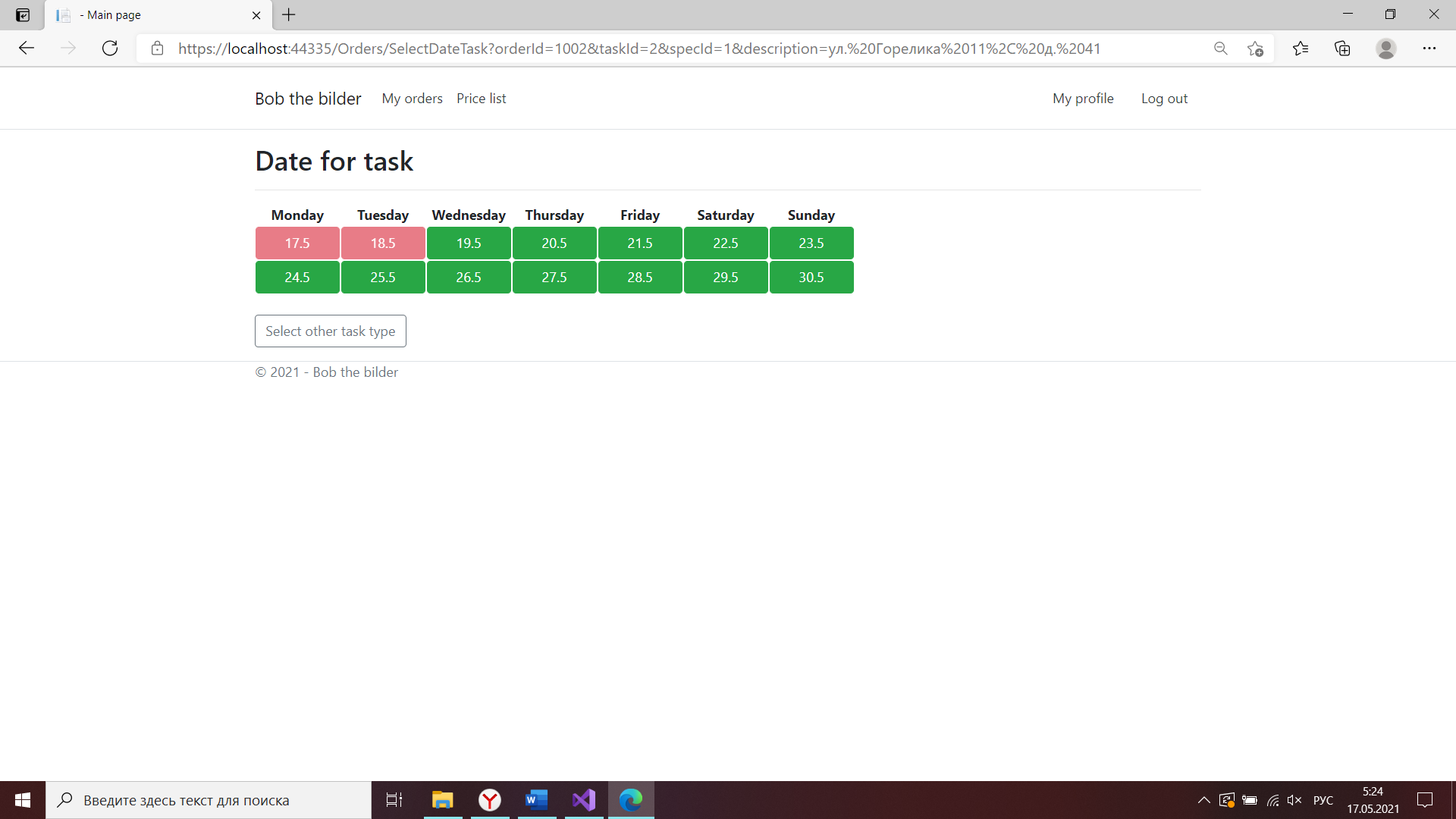


Рис 3.13 – Страница выбора даты для заказа

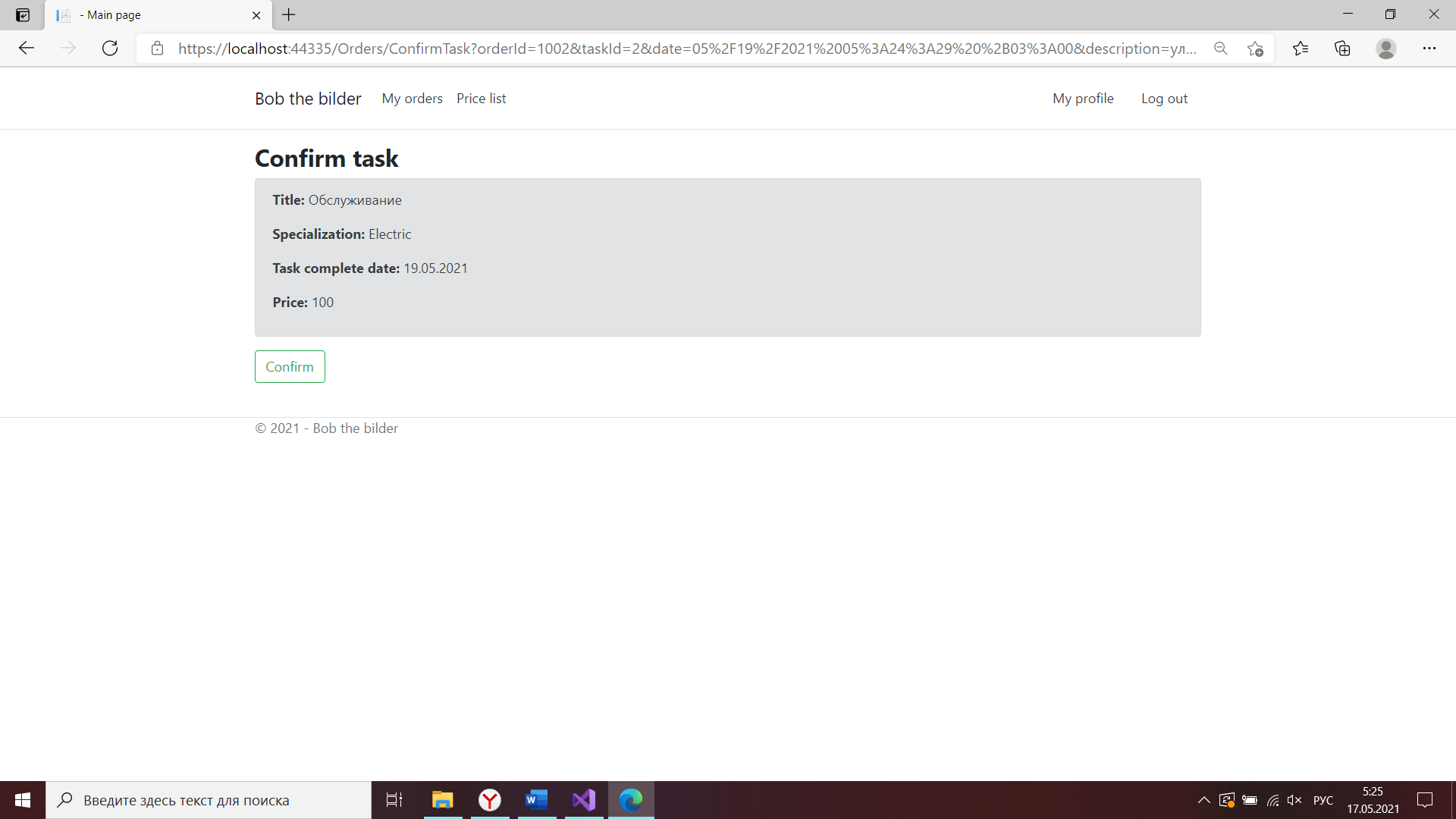


Рис 3.14 – Страница подтверждения задания для заказа

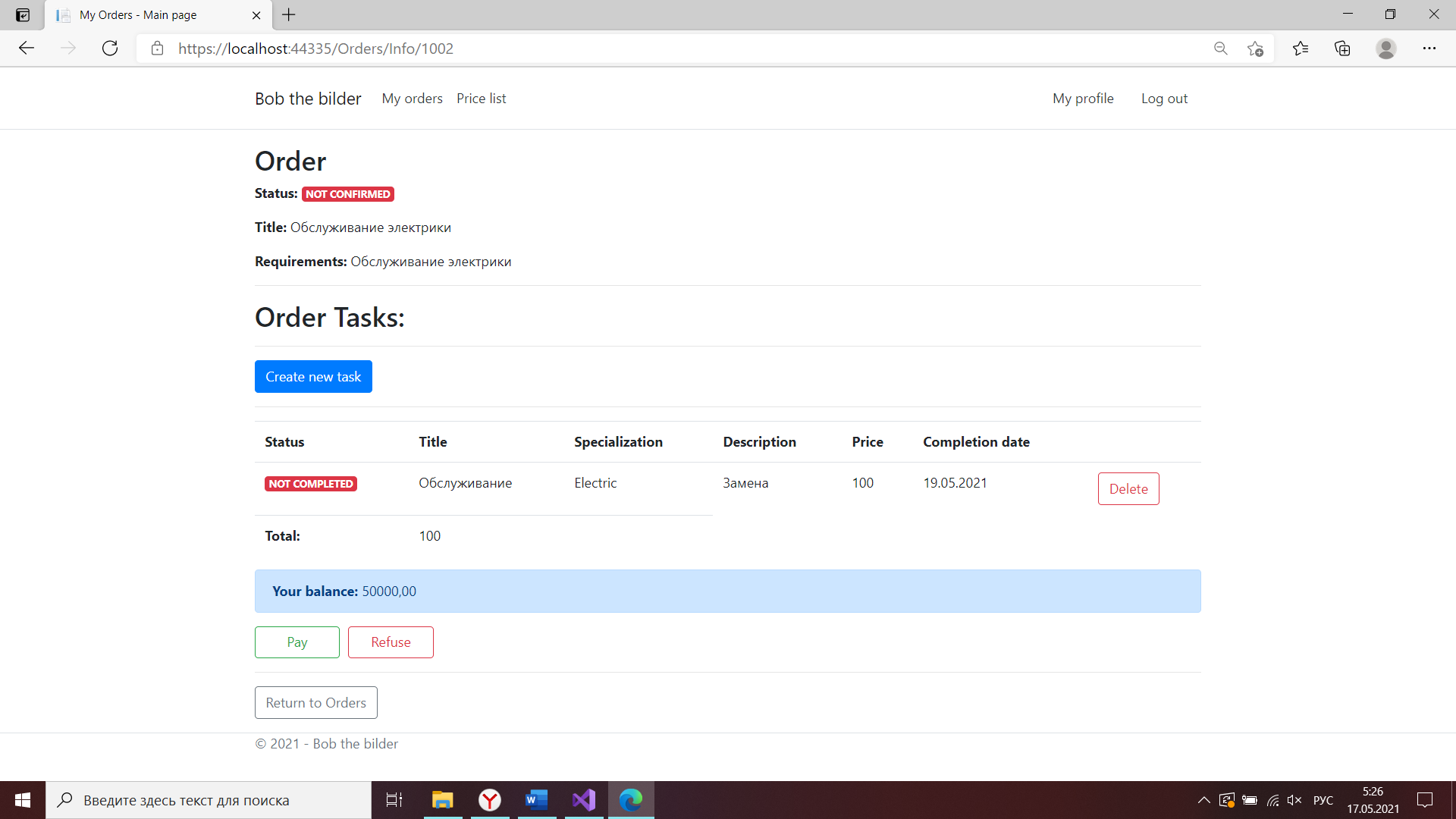


Рис 3.15 – Новое задание в заказе

Пользователь имеет возможность зайти в профиль и изменить личную инфрмацию или пополнить счет.

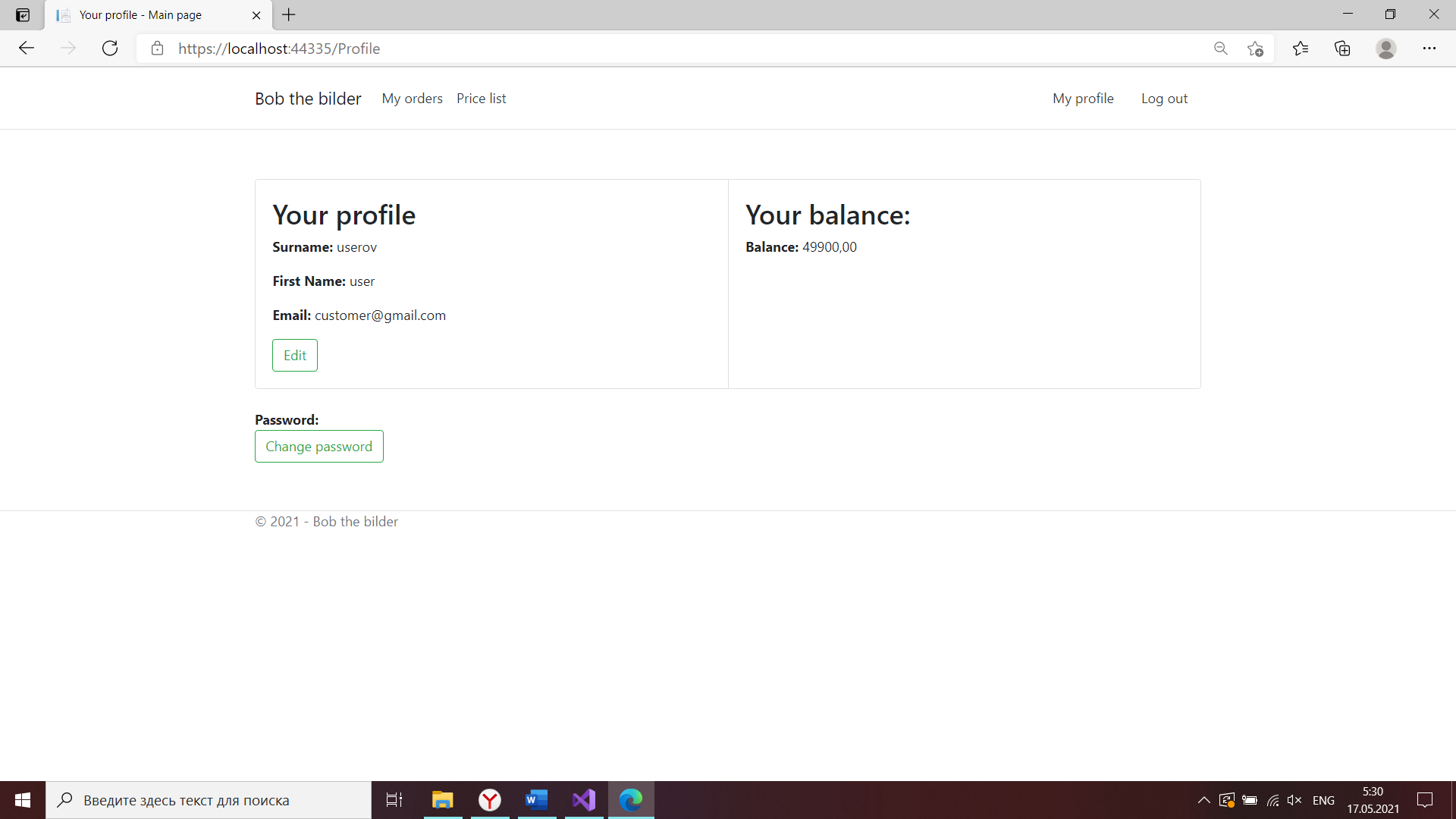


Рис 3.16 – Личная страница пользователя

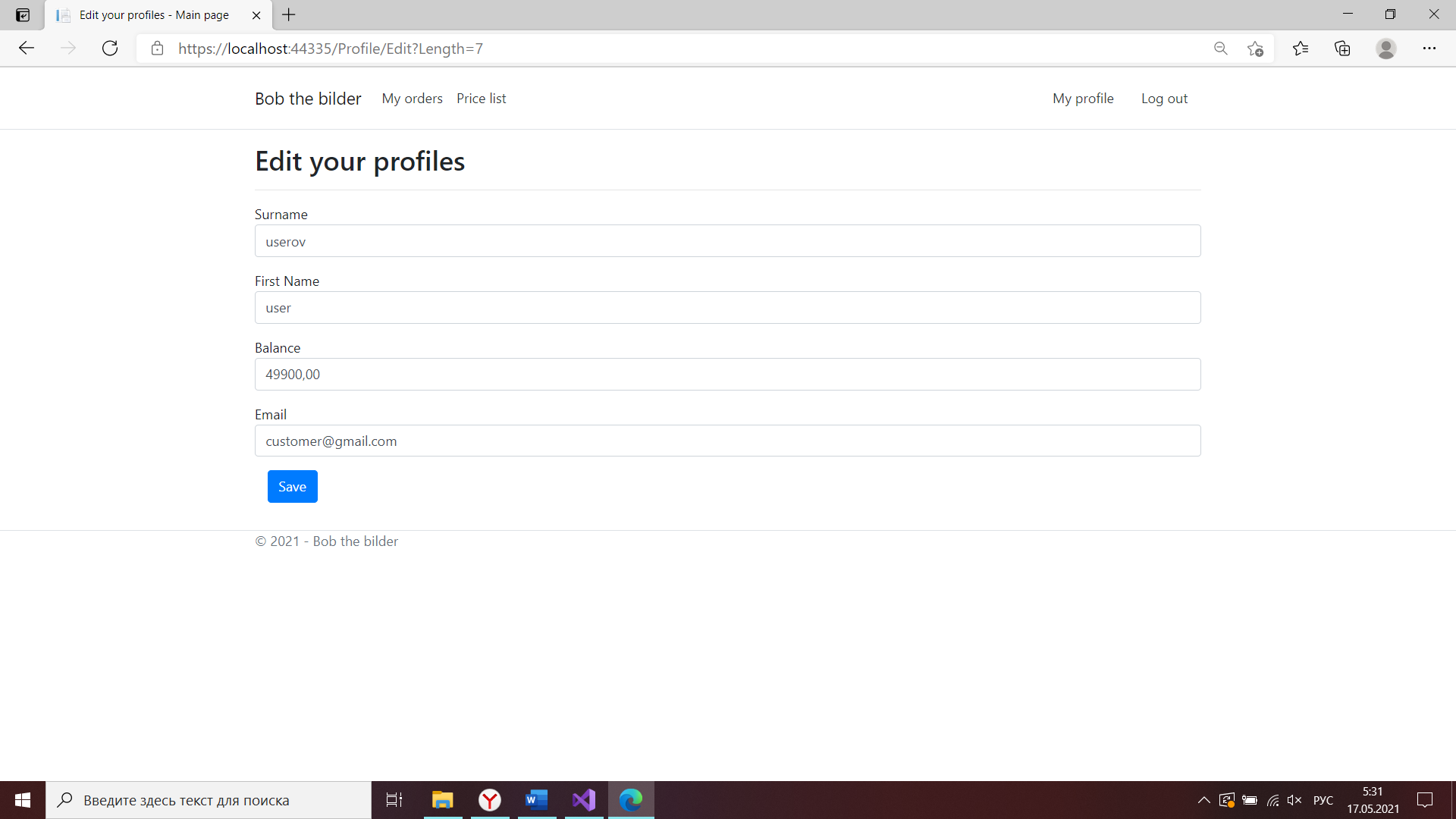


Рис 3.17 – Страница редактирования личной информации для пользователя

После пополнения счета пользователь может заплатить за составленный заказ.

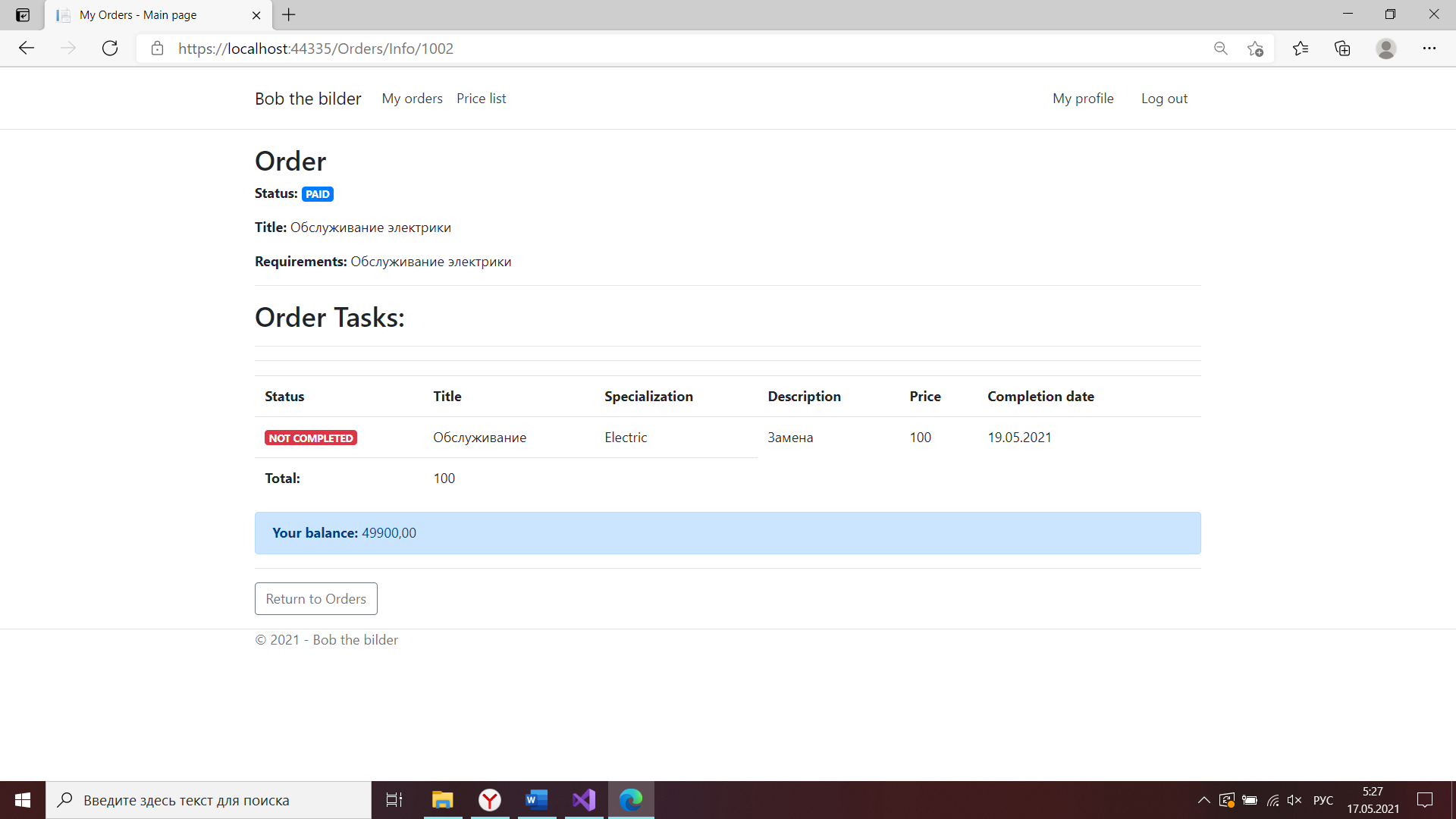


Рис 3.18 – Страница заказа после оплаты

После успешного оформления заказа, у менеджера появится новый заказ, который он может подтвердить или же отменить.

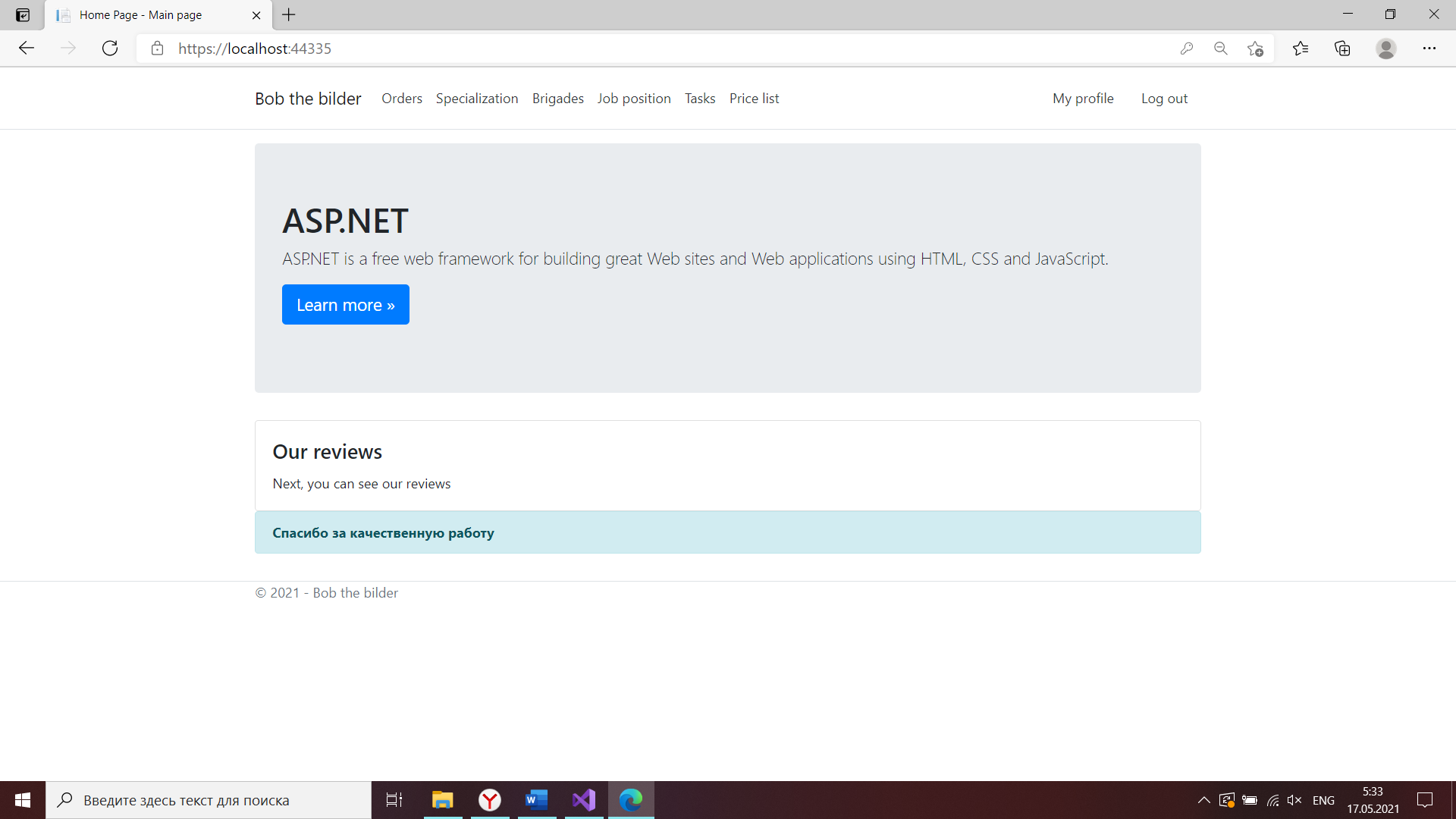


Рис 3.19 – Главная страница менеджера

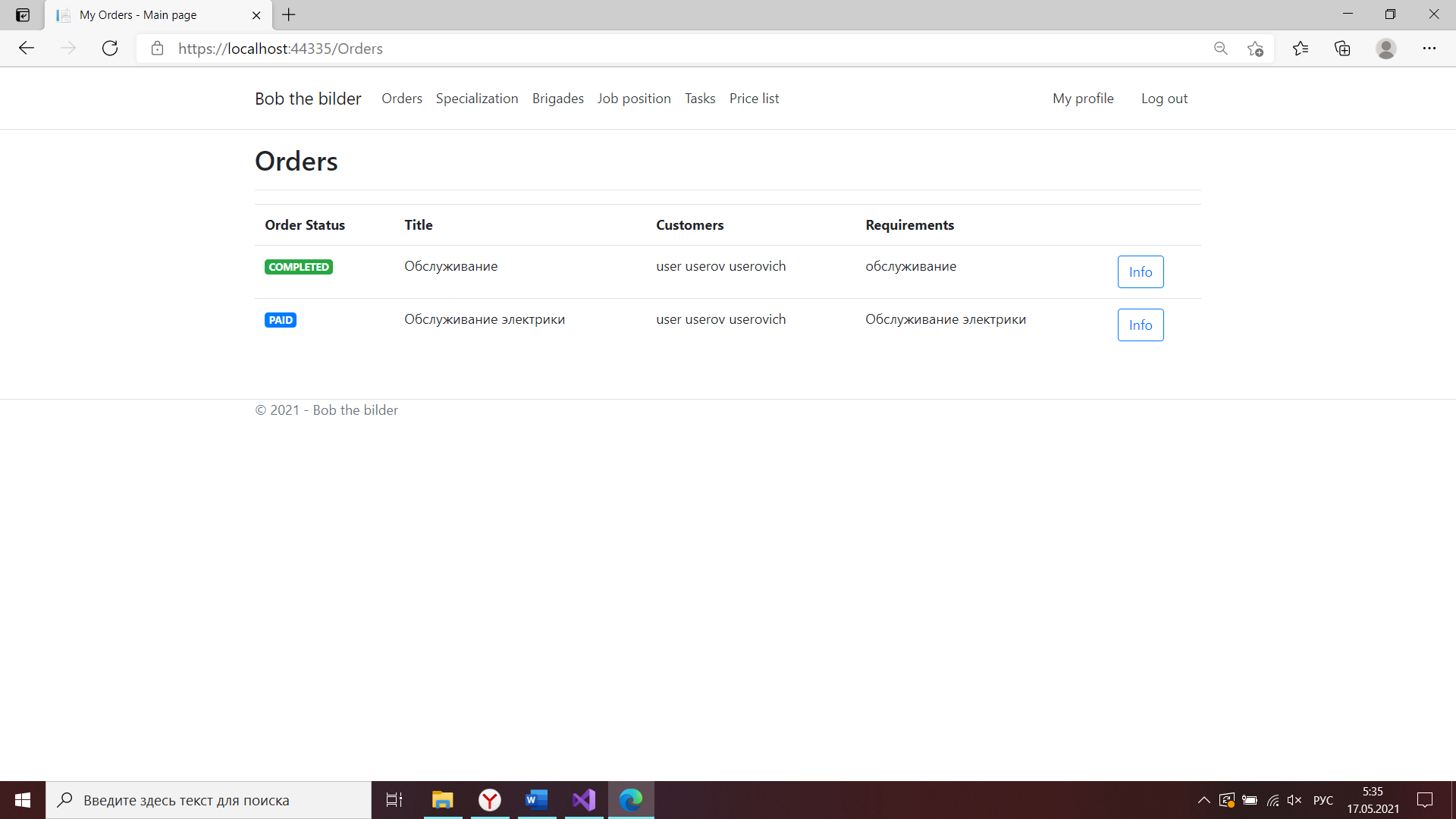


Рис 3.20 – Страница заказов менеджера

На данной странице пользователь имеет возможность узнать о подробностях заказа, где его можно подтвердить или отменить.

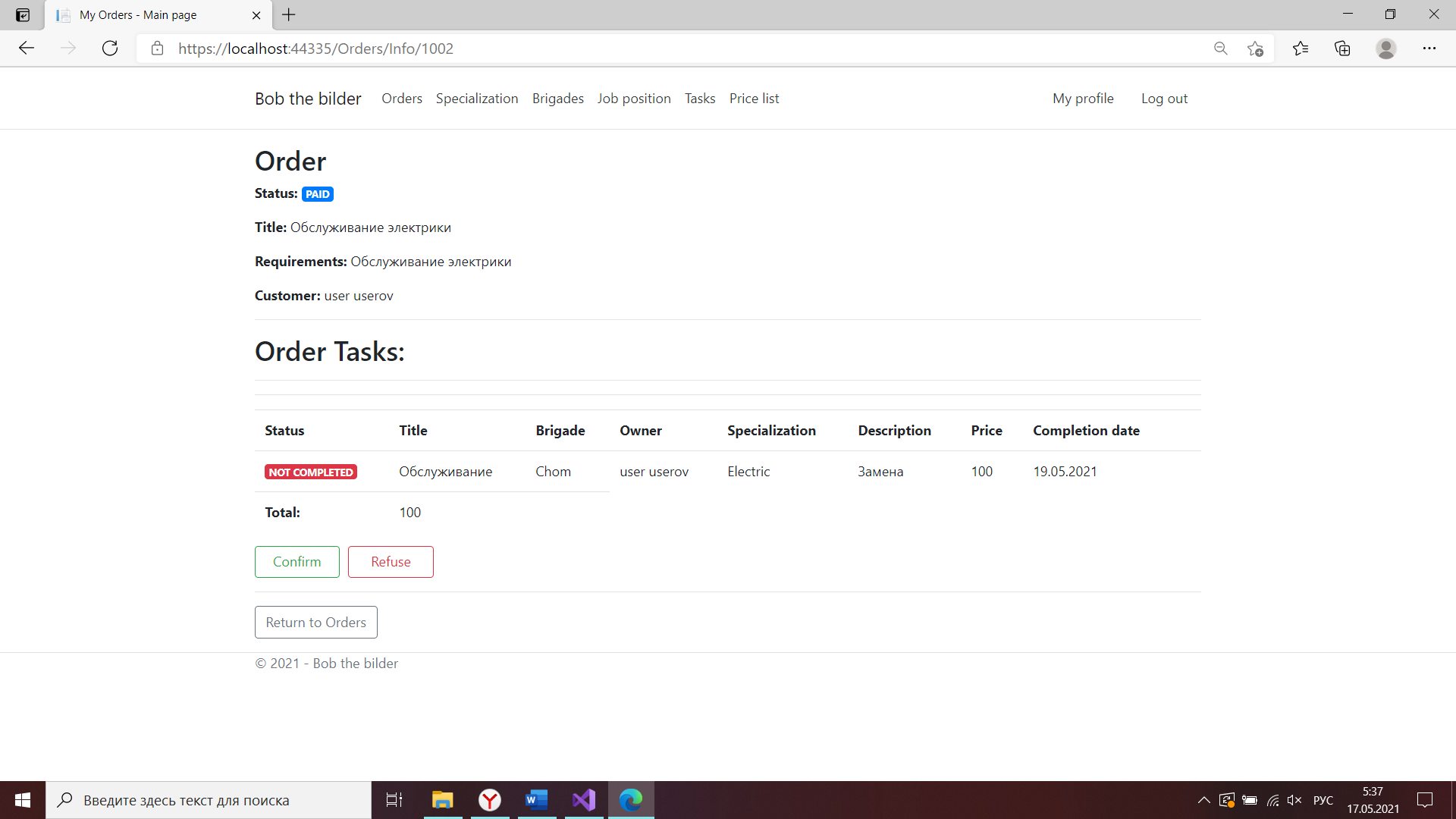


Рис 3.21 – Информация о заказе для менеджера

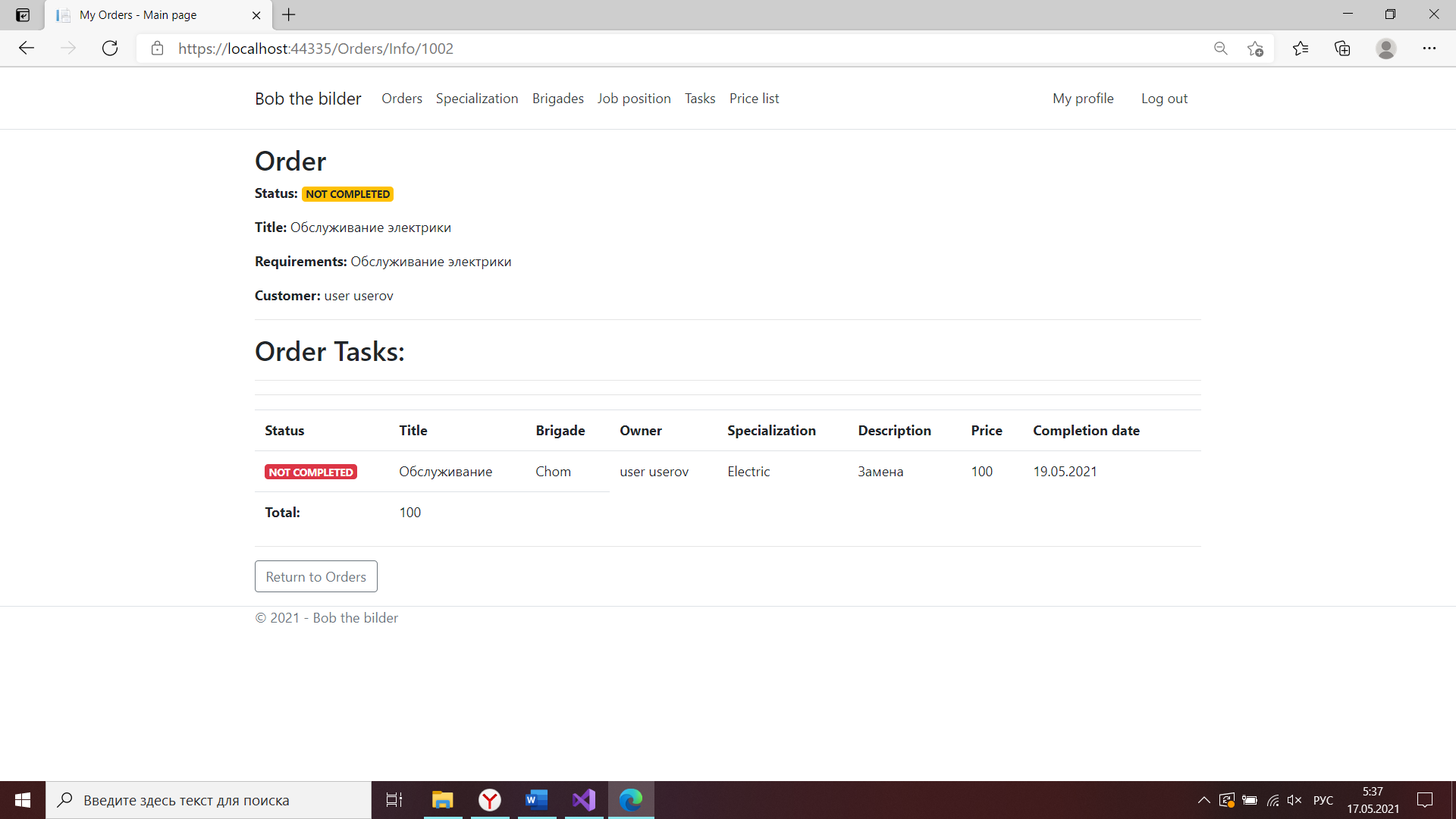


Рис 3.22 – Изменение в состоянии заказа после его подтверждения

После подтверждения заказа, у рабочих, необходимой бригады, появятся новые задания.

Менеджер имеет возможность редактировать и дополнять базу данных. Все доступные ему для редактирования таблицы видно на главной странице.

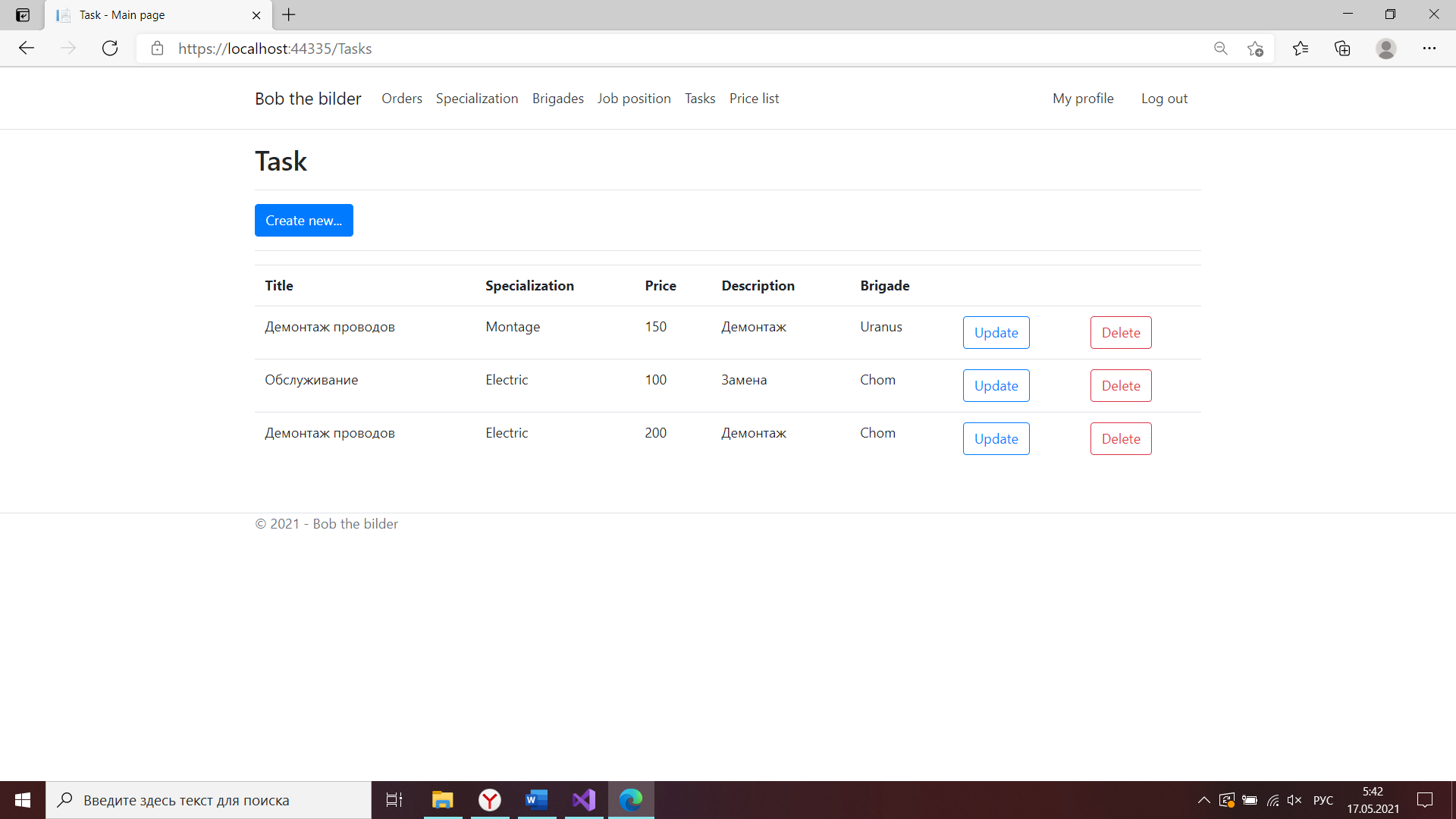


Рис 3.23 – Страница работы с заданиями

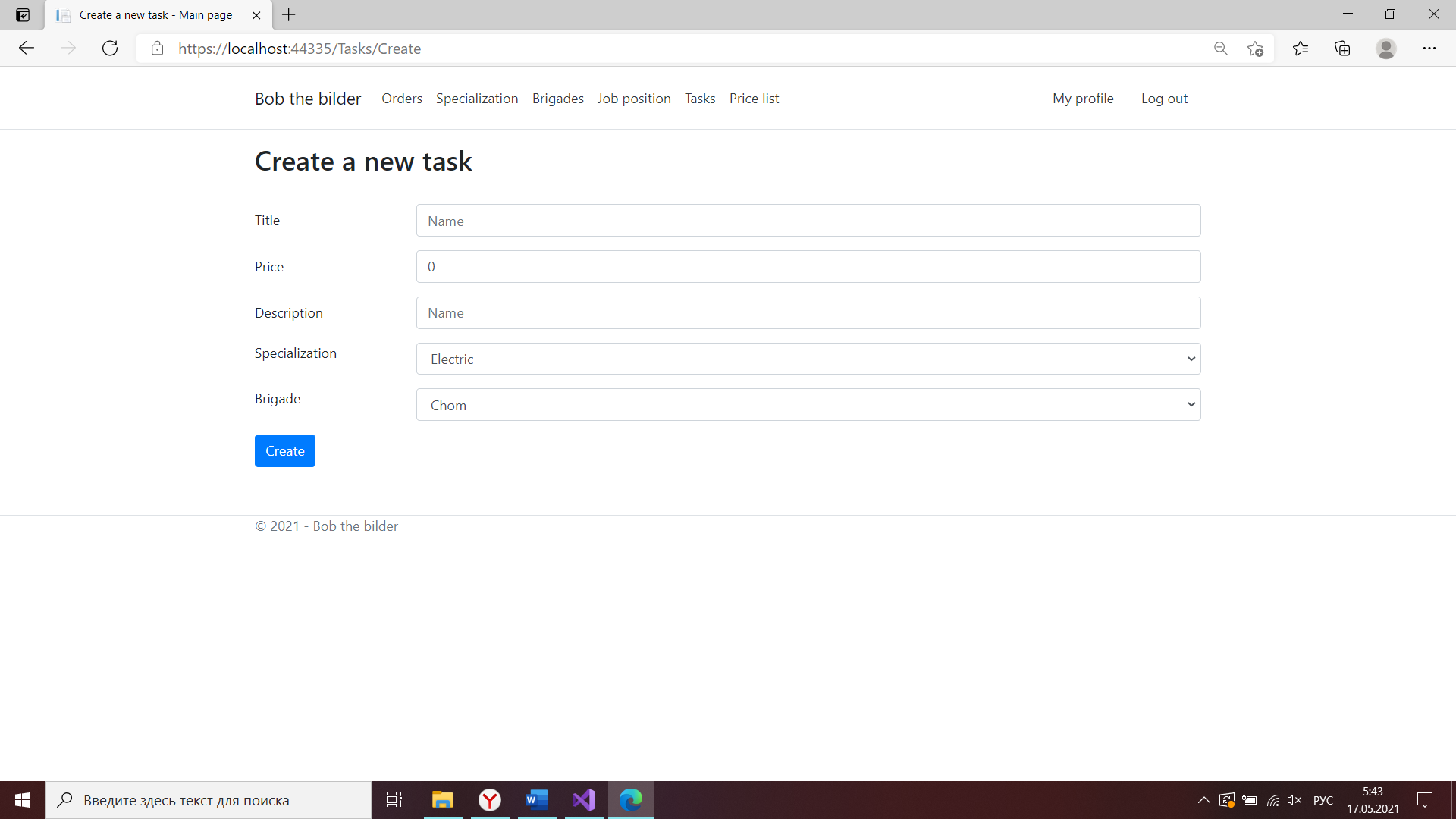


Рис 3.24 – Страница создания нового задания

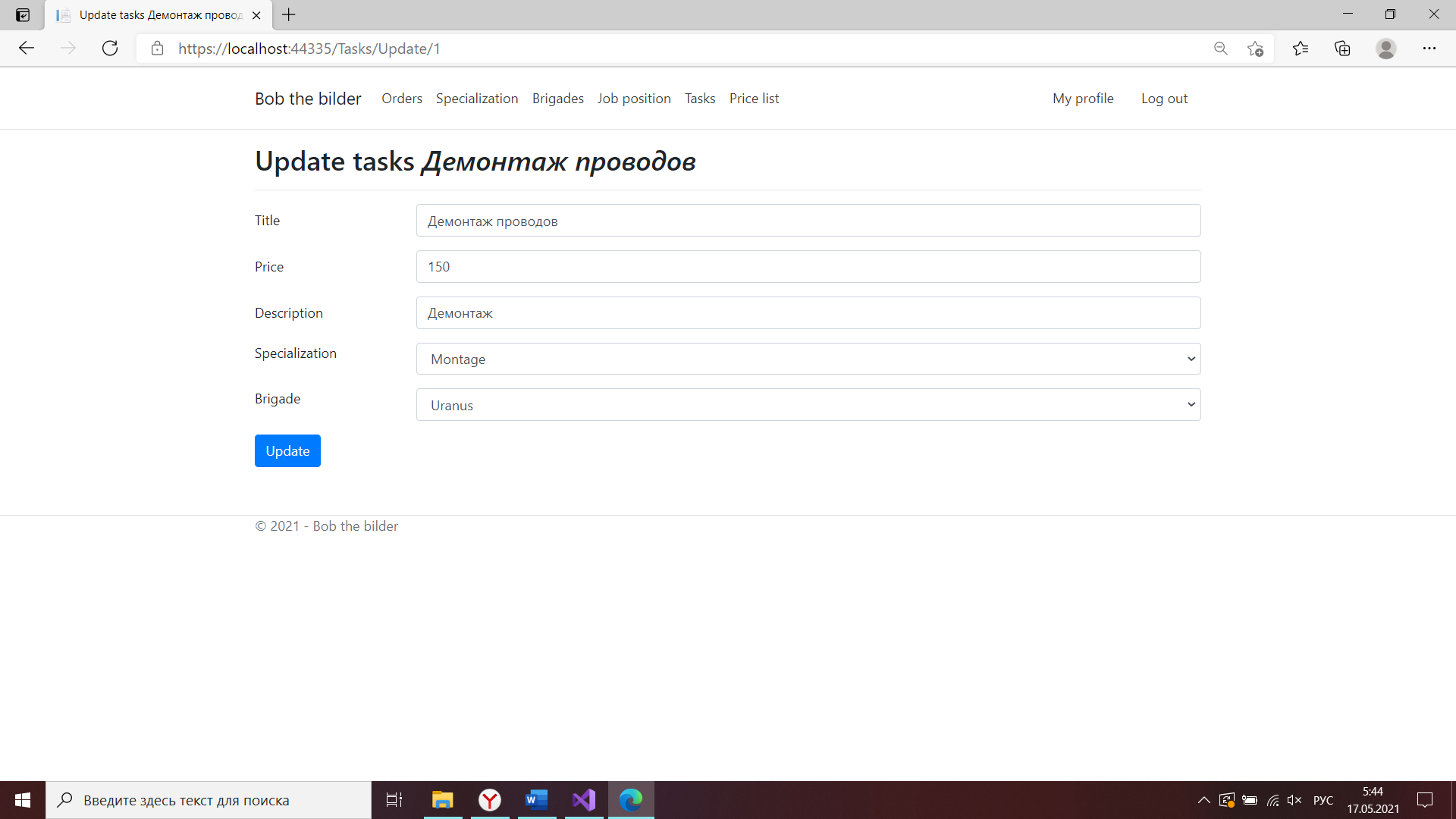


Рис 3.25 – Страница изменения задания

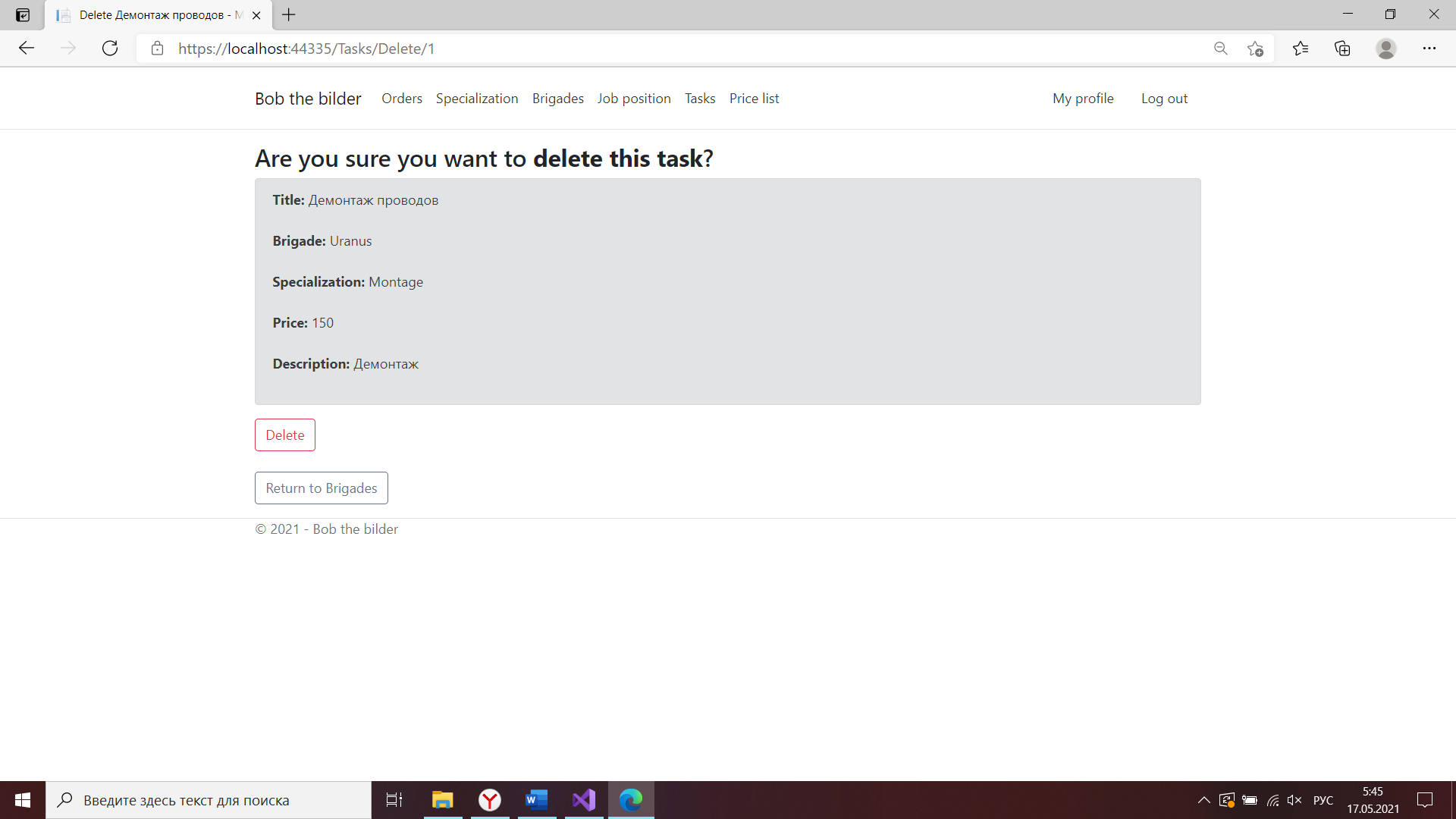


Рис 3.26 – Страница подтверждения удаления задания

Похожим образом выглядят страницы работы с специализациями, бригадами и должностями.

Главная страница рабочего выглядит следующим образом.

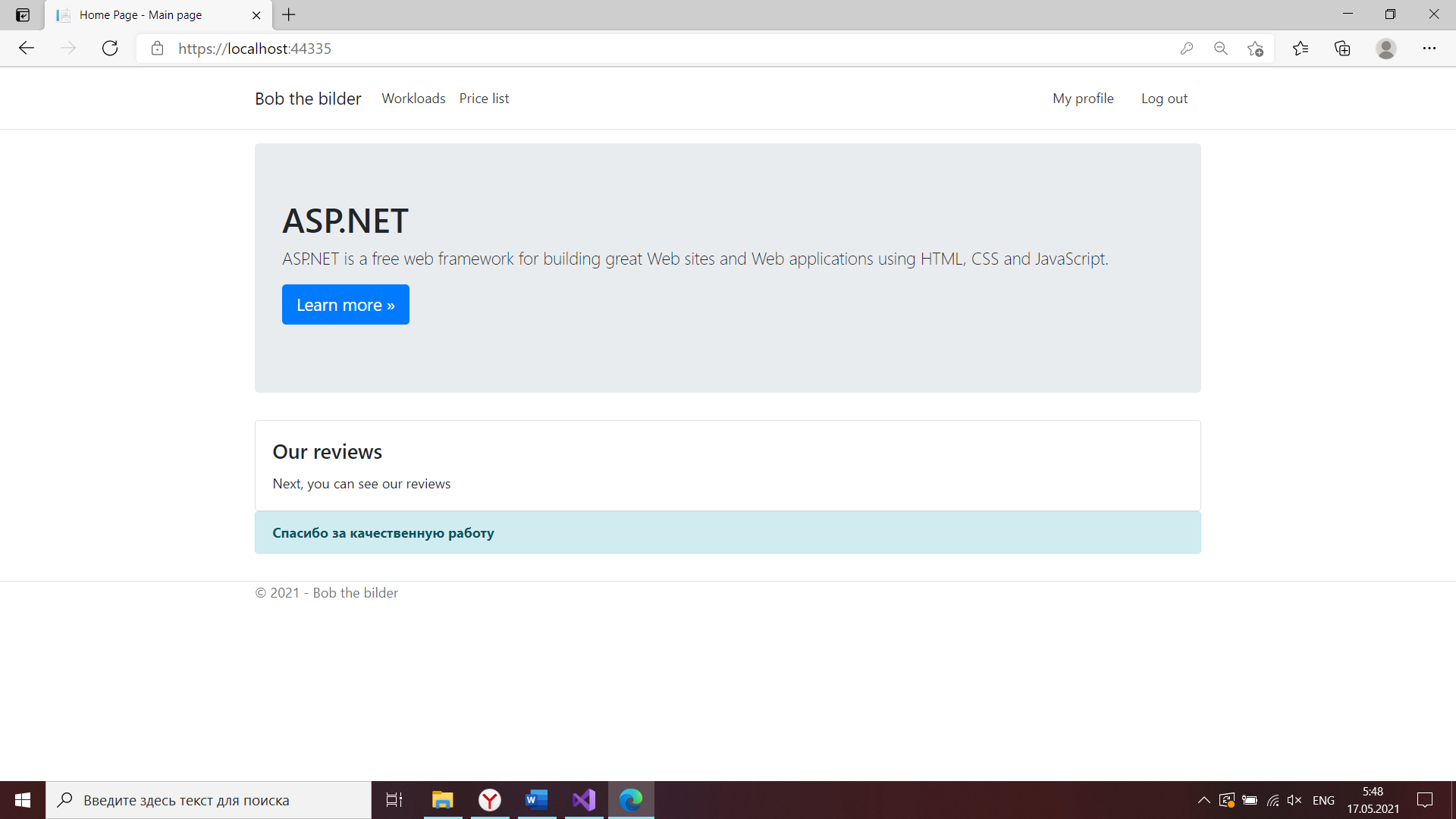


Рис 3.27 – Главная страница рабочего

С данной страницы у рабочего есть возможность перейти к странице заданий.

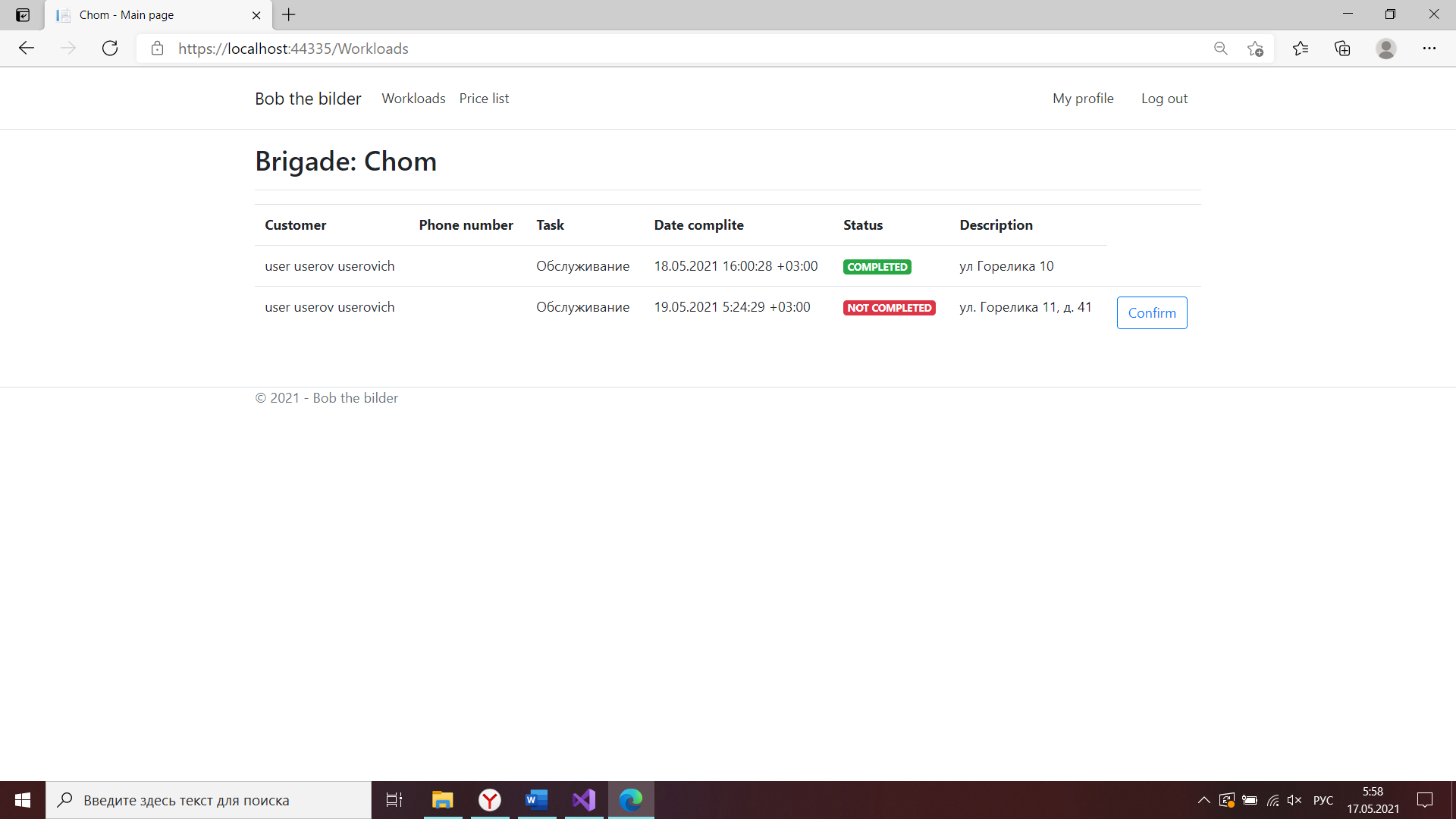


Рис 3.28 – Страница заданий рабочего

Здесь рабочий может подтвердить выполненные им задания. После подтверждения всех заданий из заказа, заказ считается выполненным.

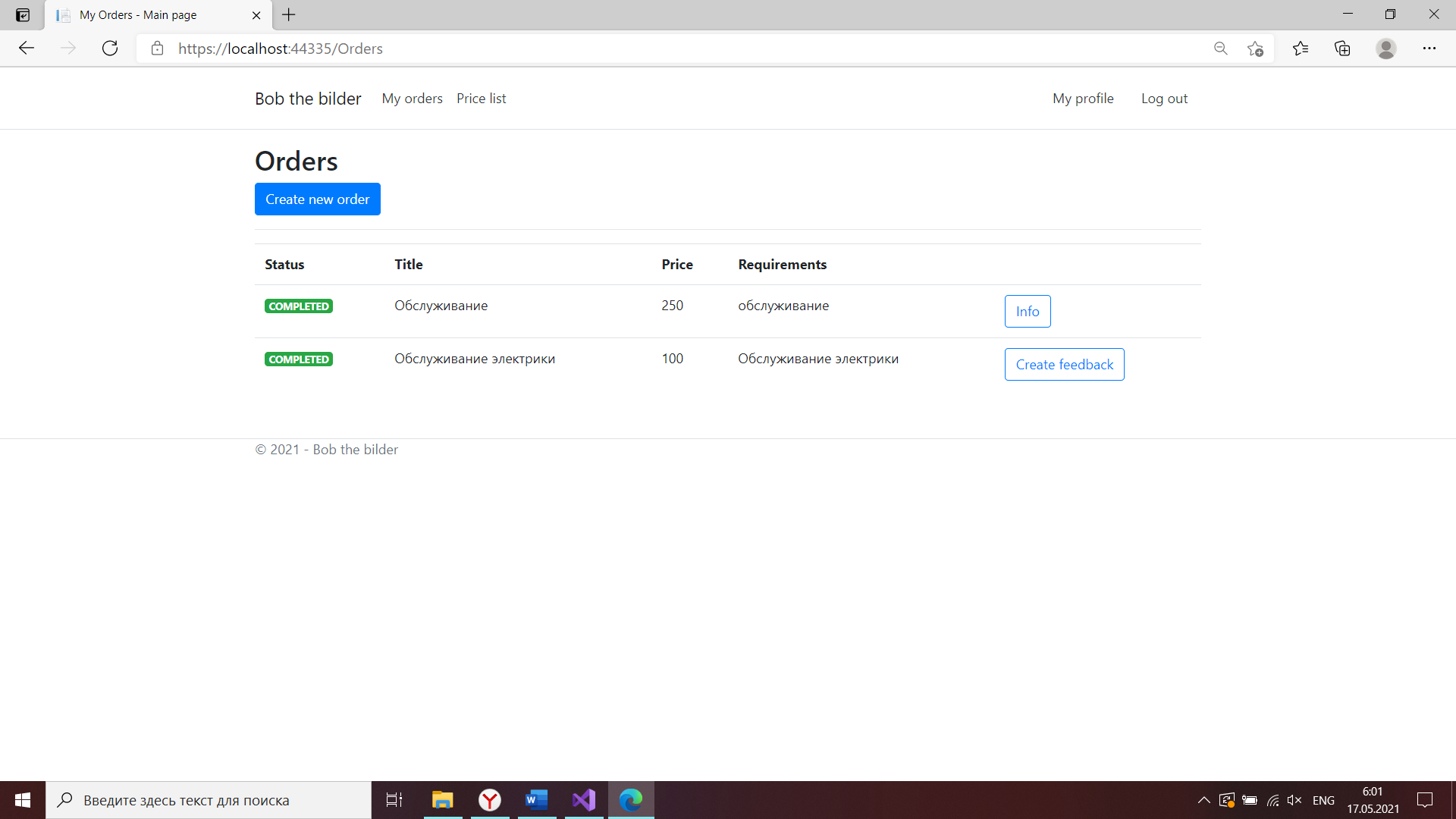


Рис 3.29 – Изменение состояния заказа после подтверждения всех заданий

Когда состояние заказа – “Complited” у пользователя появляется возможность оставить отзыв о проделанной работе, который появится на главной странице.

На главной странице администратора есть пункт “All user”, где он может просмотреть всех пользователей и, если есть необходимость, изменить роль. Также у администратора есть пункт “Statictic by task”, где он может просмотреть статистику бригад.

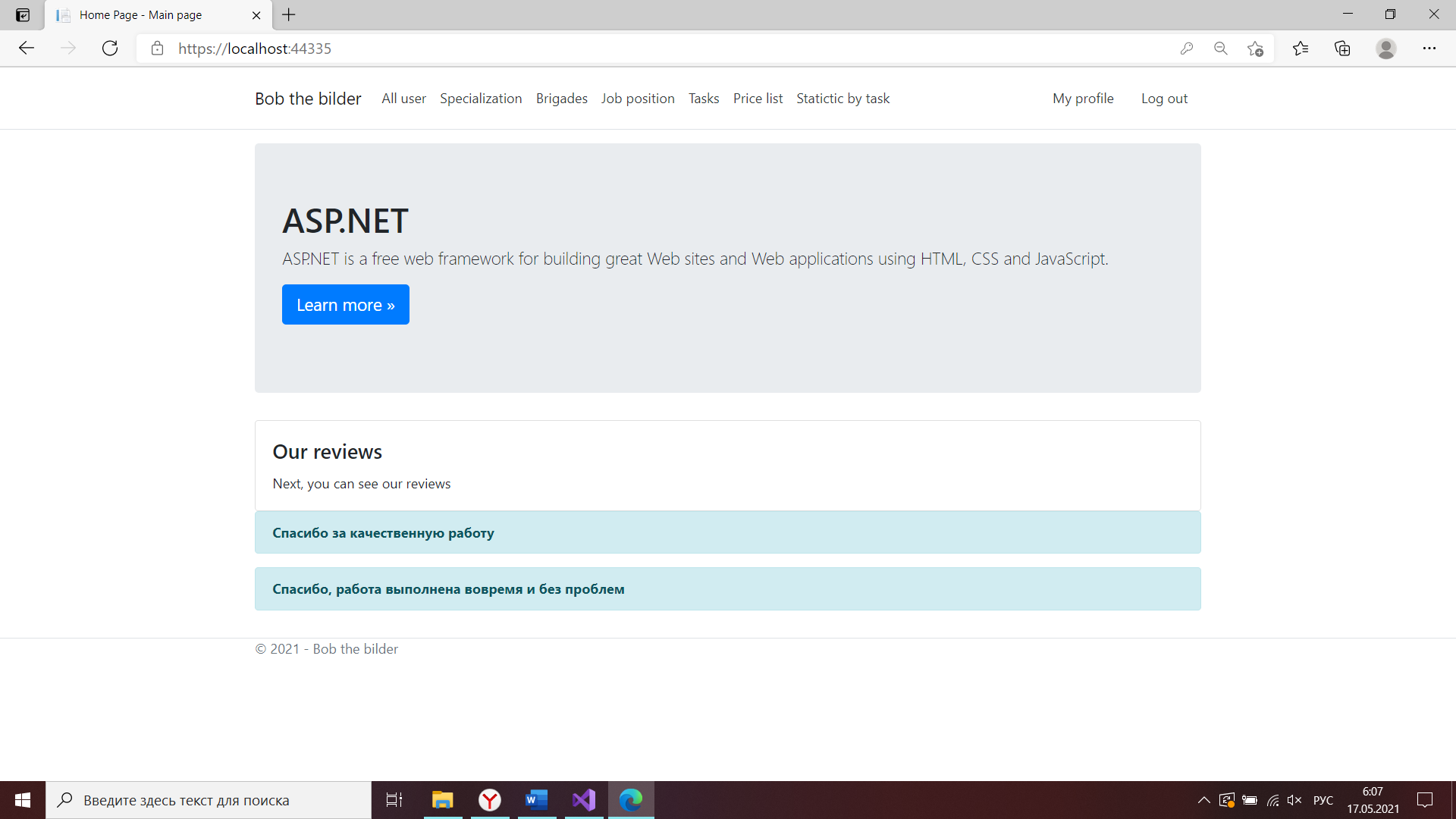


Рис 3.30 – Главная страница администратора

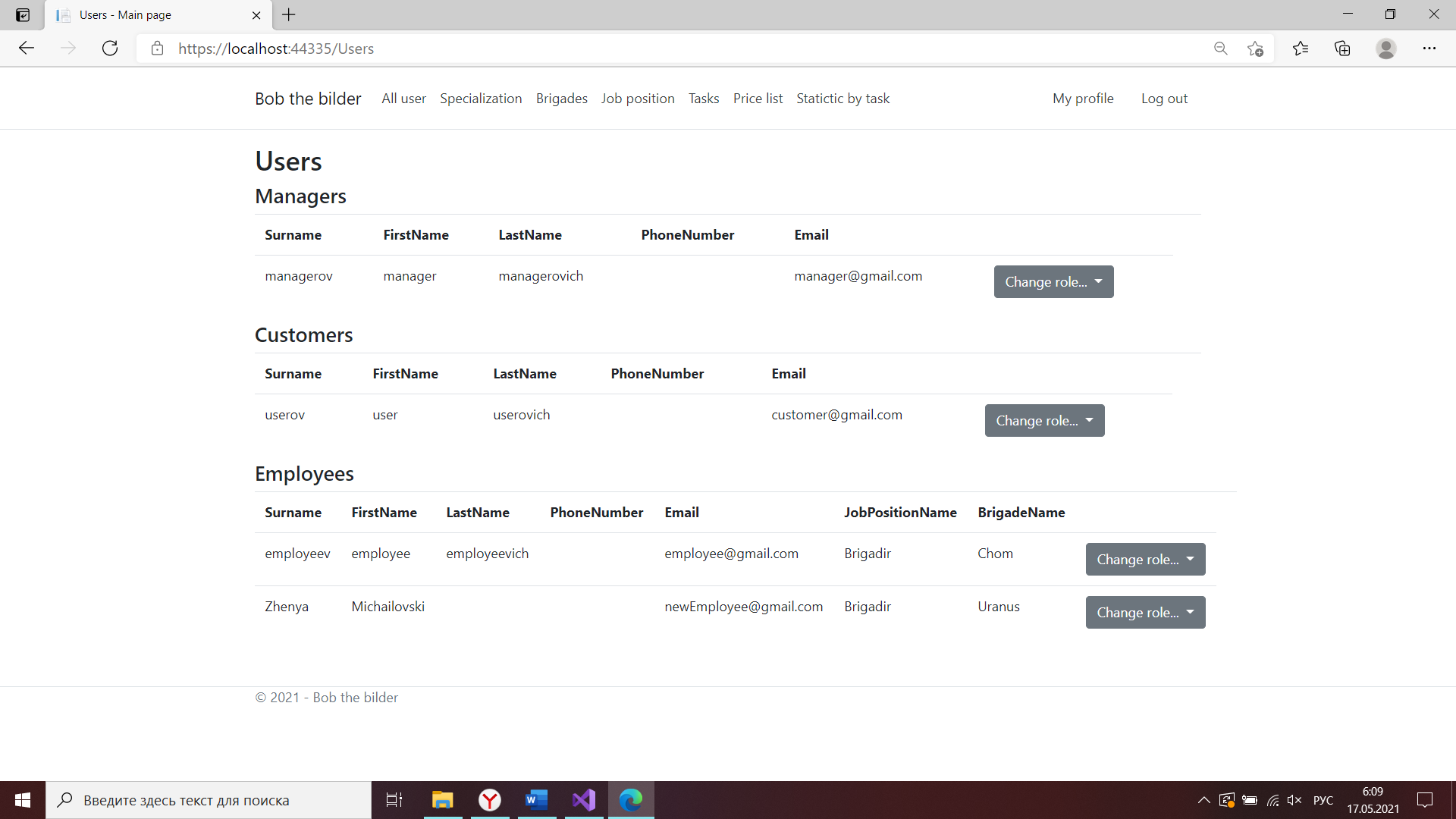


Рис 3.31 – Страница всех пользователей

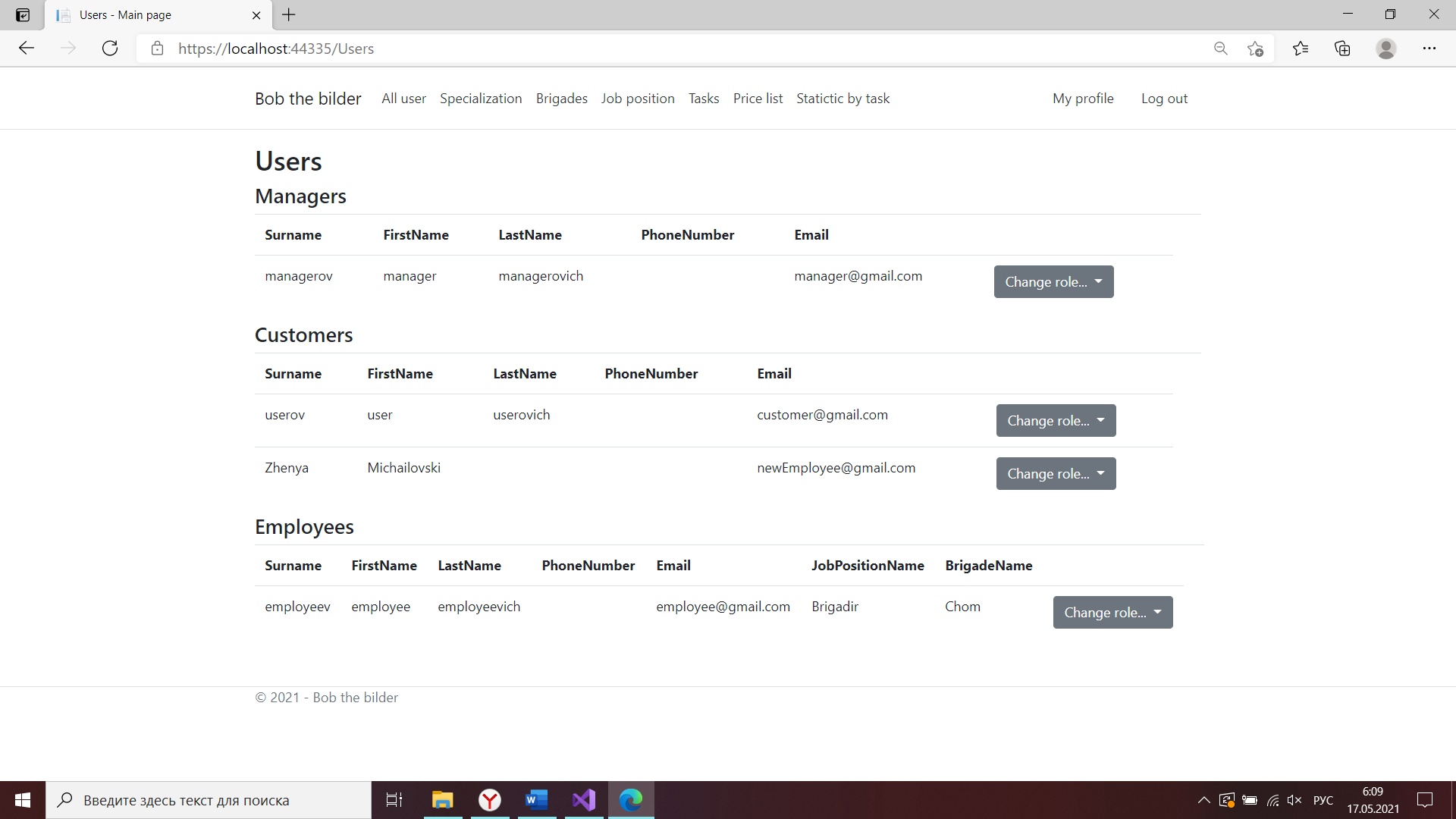


Рис 3.32 – Обновленная таблица пользователей

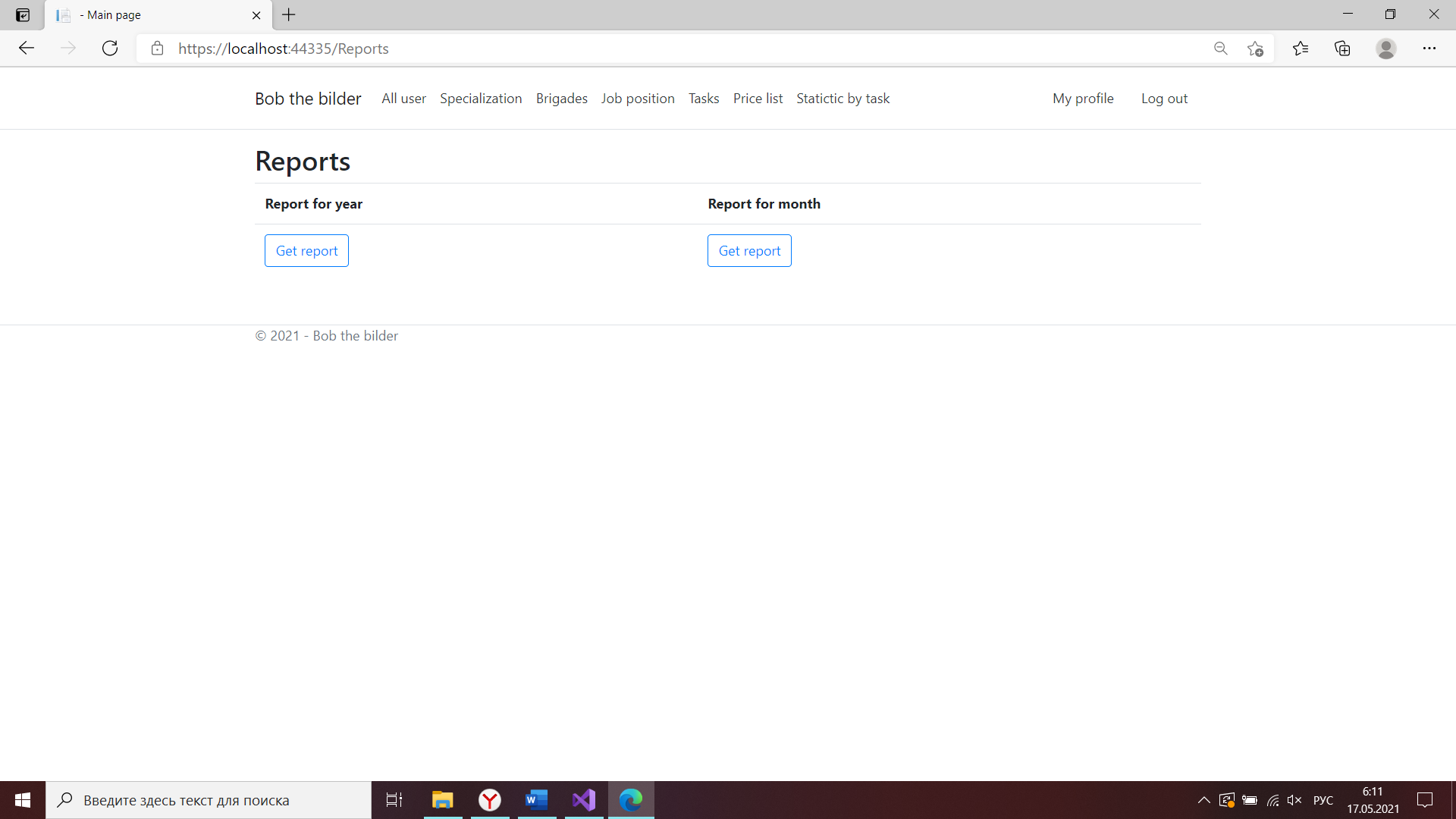


Рис 3.33 – Страница статистики

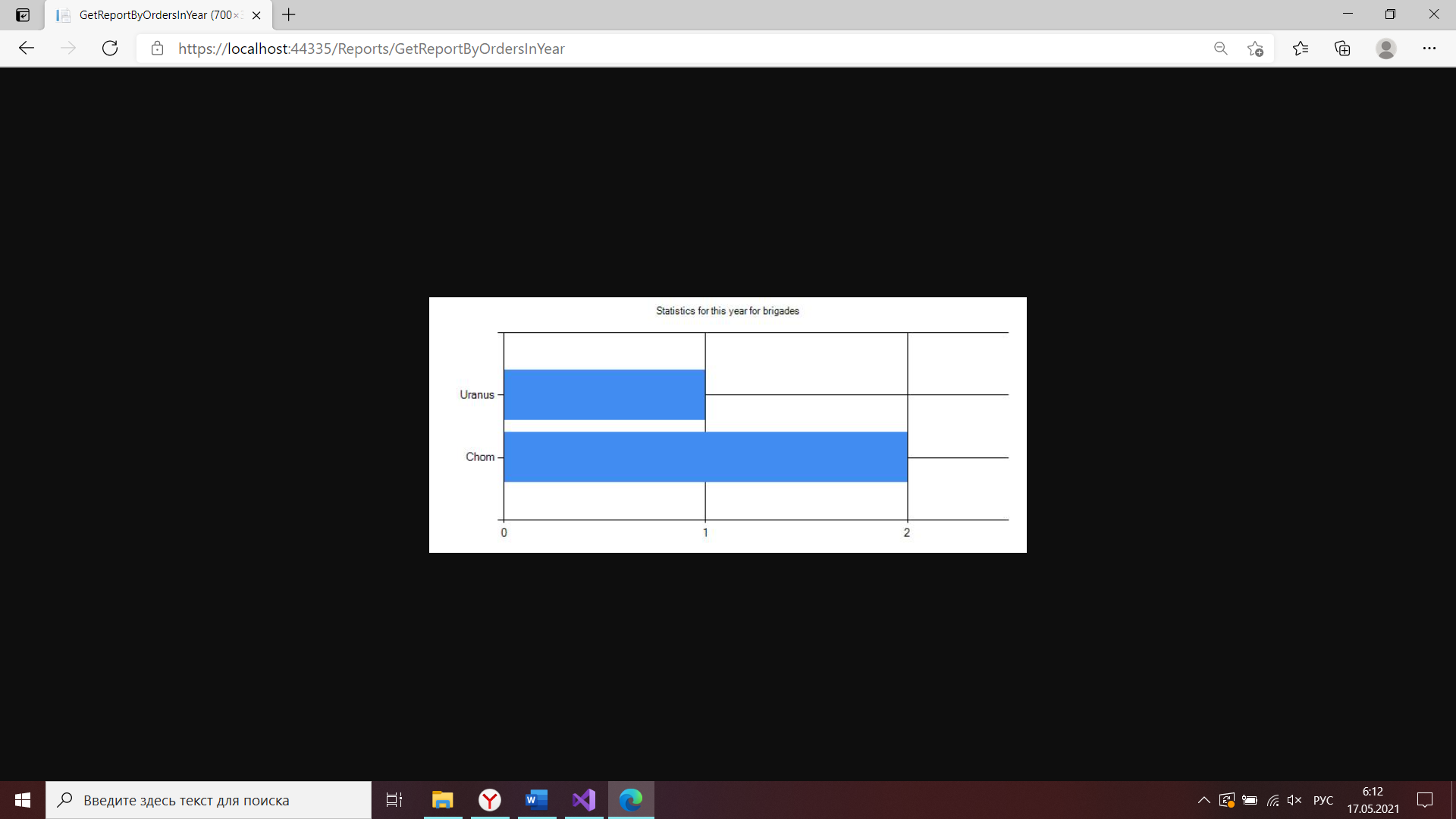


Рис 3.34 – Статистика за год

1. **ТЕСТИРОВАНИЕ**
   1. **Верификация**

В качестве примера верификации будет рассмотрена функция создания заказа, добавления в него заданий и его оплаты.

При нажатии на кнопку создания заказа, приложению необходимо сделать запросы в базу данных по специальностям, бригадам и заказам, чтобы подготовить данные для заполнения.

Для выбора специальности, программе необходимо выгрузить из базы данных список всех специальностей.

Далее, на выборе типа задания, программе необходимо отфильтровать задания, по выбранной специальности.

При выборе даты проведения задания, программе необходимо отфильтровать свободные и занятые дни бригады так, чтобы если в какой-то день у бригады уже есть задание, то кнопка выбора этой даты была неактивна.

Как видно из рисунков 3.7 – 3.14 программа справляется с поставленной задачей. Таким образом, тест прошел успешно.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта, было создано веб-приложение на базе ASP.NET, которое соответствует всем заявленным требованиям и нуждам автоматизации. Была изучена предментая область и спроектированна база данных. Была освоена система управления реляционными базами данных MS SQL Server, а также принципы работы с технологией ASP.NET.

В разработанном приложении были учтены нужны его будующих пользователей и обработаны многие исключения. У проекта есть возможные вектора развития и дополнения функционала, которые можно реализовать при более длительной и углубленной проработке приложения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Джеффри Рихтер. - М.: Питер, 2013. - 928 c
2. Джозеф Альбахари и Бен Альбахари. C# 7.0. Карманный справочник. – «O’Reilly Media, Inc», 2017. – 156 с.
3. Эндрю Троелсен. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание. — М.: «Вильямс», 2013. — 792 с.
4. Microsoft.Office.Interop.Word Namespace [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word?view=word-pia, свободный. – Загл. с экрана.
5. Отображение значения Enum в C# на русском [Электронный ресурс]. – URL: https://shwanoff.ru/enum-to-string/, свободный. – Загл. с экрана.