**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«Национальный исследовательский университет ИТМО»**

**(Университет ИТМО)**

**Факультет: Инфокоммуникационных технологий**

**Образовательная программа: Инфокоммуникационные системы**

**Направление подготовки (специальность): Прикладное программирование в инфокоммуникационных системах**

О Т Ч Ё Т

по лабораторным работам

по дисциплине «Создание ПО инфокоммуникационных систем»

Выполнили: **студенты группы K34211**

**Дарзнек Лариса Викторовна**

**Логвинов Лев Анатольевич**

Проверил: **Осипов Никита Алексеевич**

Санкт-Петербург

2023

## Цели работы

Создать web-приложение, реализующее запросы для доступа к данным.

## Выбор средств разработки веб-приложения

Для разработки backend системы было решено использовать ASP.NET Core MVC, который является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету.

Была построена диаграмма базы данных разрабатываемого приложения – рисунок 1.

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Рисунок 1 – Диаграмма БД

## Реализация веб-приложения

Систему было решено разделить на 4 проекта (рисунок 2), каждый из которых отвечает за определенную область приложения, однако все вместе они образуют систему. Подробнее о каждом проекте:

* Bulochka.DataAccess - содержит классы для работы с данными и взаимодействия с базой данных;
* Bulochka.Models - содержит модели системы;
* Bulochka.Utility - содержит дополнительные классы и перечисления необходимые для работы системы;
* Bulochka.Web - содержит контроллеры и представления системы отдельно для администратора и клиента, а также конфигурационные файлы, html, css, js- файлы.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Проекты приложения

Класс ApplicationDbContext (рисунок 3) унаследован от базового класса IdentityDbContext, который позволяет взаимодействовать с базой данных (свойства DbSet<T> позволяют работать с отдельными таблицами).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - Класс ApplicationDbContext

Для реализации системы было решено использовать паттерн репозиторий. Репозитории — это классы или компоненты, которые содержат логику, необходимую для доступа к источникам данных. Репозитории предоставляют централизованные функции для доступа к данным, позволяя улучшить управление и отделить инфраструктуру или технологию, используемую для доступа к данным, от модели предметной области.

Рассмотрим реализацию данного паттерна в приложении. Есть общий интерфейс для создания репозиториев – IRepository (рисунок 4). В нем определены методы, необходимы для работы с данными (например добавление или получение всех объектов)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Интерфейс IRepository

Далее есть класс, реализующий этот интерфейс, то есть в нем реализована общая логика методов интерфейса – Repository (рисунок 5). Например, метод Add, в котором реализована логика добавления объекта в соответствующую базу данных.

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Рисунок 5 – Класс Repository

Репозитории отдельных сущностей наследуются от базового класса Repository с указанием класса модели, а также реализуют интерфейс отдельно созданного интерфейса репозитория для конкретной сущности. Также они могут иметь дополнительные методы, расширяющие функционал. Пример представлен на рисунке 6.

Изображение выглядит как текст, снимок экрана, внутренний, экран

Автоматически созданное описание

Рисунок 6 – Класс ProductRepository

Также был добавлен интерфейс IUnitOfWork (рисунок 7) и реализующий его класс, являющийся входной точкой для получения доступа ко всем добавленным в него репозиториям. Здесь определен метод Save, позволяющий сохранить все изменения базы данных.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Интерфейс IUnitOfWork

В проекте Bulochka.Models содержатся модели, необходимы для работы системы. Рассмотрим одну из них – CompanyBranch или филиал компании (рисунок 8). Модель включает поля Id (автоматически первичный ключ), City, StreetAdress, PostalCode, PhoneNumber (может принимать значение null).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Модель филиала компании

После создания модели, в класс ApplicationDbContext добавляется свойство DbSet<T>, где Т – это класс модели и добавляется миграция. Миграция генерирует класс, при выполнении которого обновляется база данных (в данном случае добавилась таблица CompanyBranch с указанными полями).

Проект Bulochka.Utility нет причин расписывать подробно, так как там просто содержатся статические классы, необходимые для работы системы.

Проект Bulochka.Web последний в списке, но далеко не последний по значению. Тут находятся не только контроллеры и представления, но и стартовый файл, который запускает проект, а также файл с настройками проекта, в котором указан путь доступа к базе данных и множество других полезных файлов.

Как видно на рисунке 9, есть папка Areas, в которой есть папки Admin, Customer, в каждой из которых лежат контроллеры и представления, с которыми можно взаимодействовать только в этих областях. Например, логика создание продукта находится в файле ProductController, который расположен в области Admin. Такая логика используется для того, чтобы только администратор мог работать с продуктами (добавлять, удалять их и т. д.). В папке Views, лежащей в корневом каталоге лежат общие представления, на основе которых строятся страницы. Наконец в папке Identity находятся автоматически сгенерированные файлы для работы с регистрацией, авторизацией пользователей.

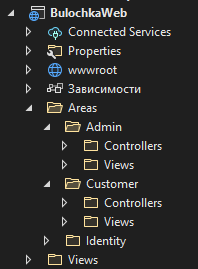


Рисунок 9 – Иерархия проекта

## Обзор работы веб-приложения и тестирование функционала

Регистрация в системе. Пользователю представлена возможность зарегистрироваться в системе. Нужно указать полное имя, email, номер телефона, адрес, а также придумать пароль. В качестве модели пользователя использовалась стандартная модель пользователя от Microsoft дополненная несколькими полями.

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

Рисунок 10 – Страница регистрации

После нажатия на кнопку «регистрация» приложение проверяет корректность введенных данных, и, если ошибок нет, добавляет запись о пользователе в базу данных и возвращает пользователя на главный экран. При такой регистрации пользователь имеет роль «Клиент», а поэтому его возможности в приложении ограничены просмотром контента.

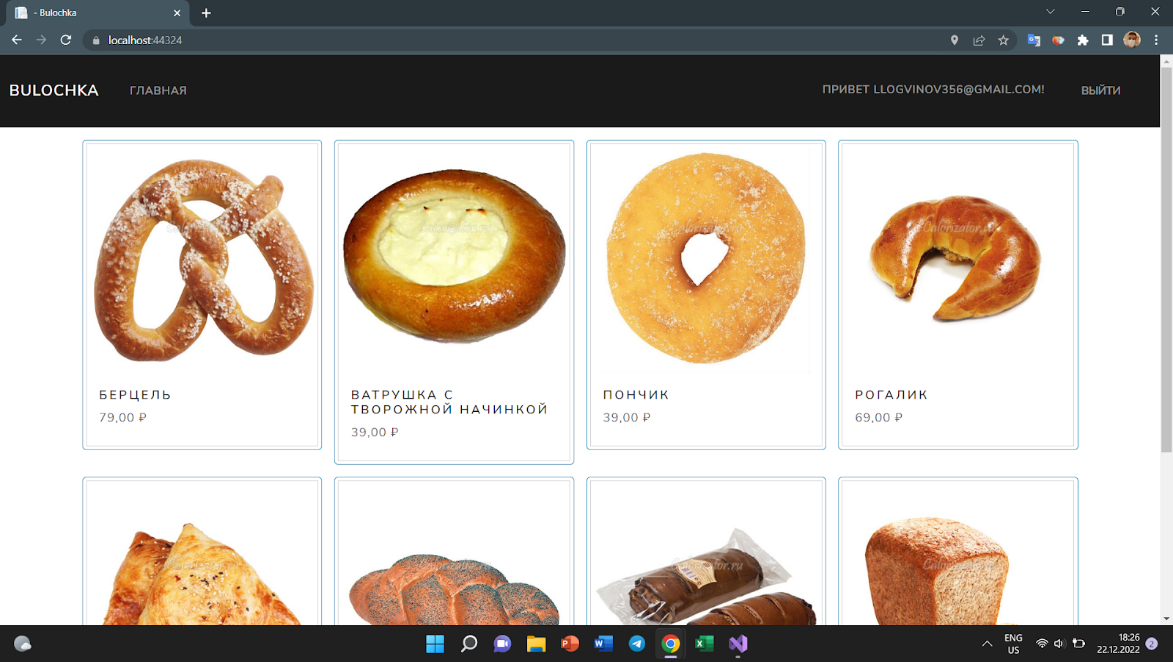


Рисунок 11 – Каталог товаров

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Рисунок 12 – Дополнительная информация о товаре

Администратор. Был осуществлен вход в систему пользователем с правами «Администратор». Теперь в верхнем меню появился еще один элемент, позволяющий управлять контентом. Можно управлять товарами, филиалами и добавлять пользователей, указывая роли.

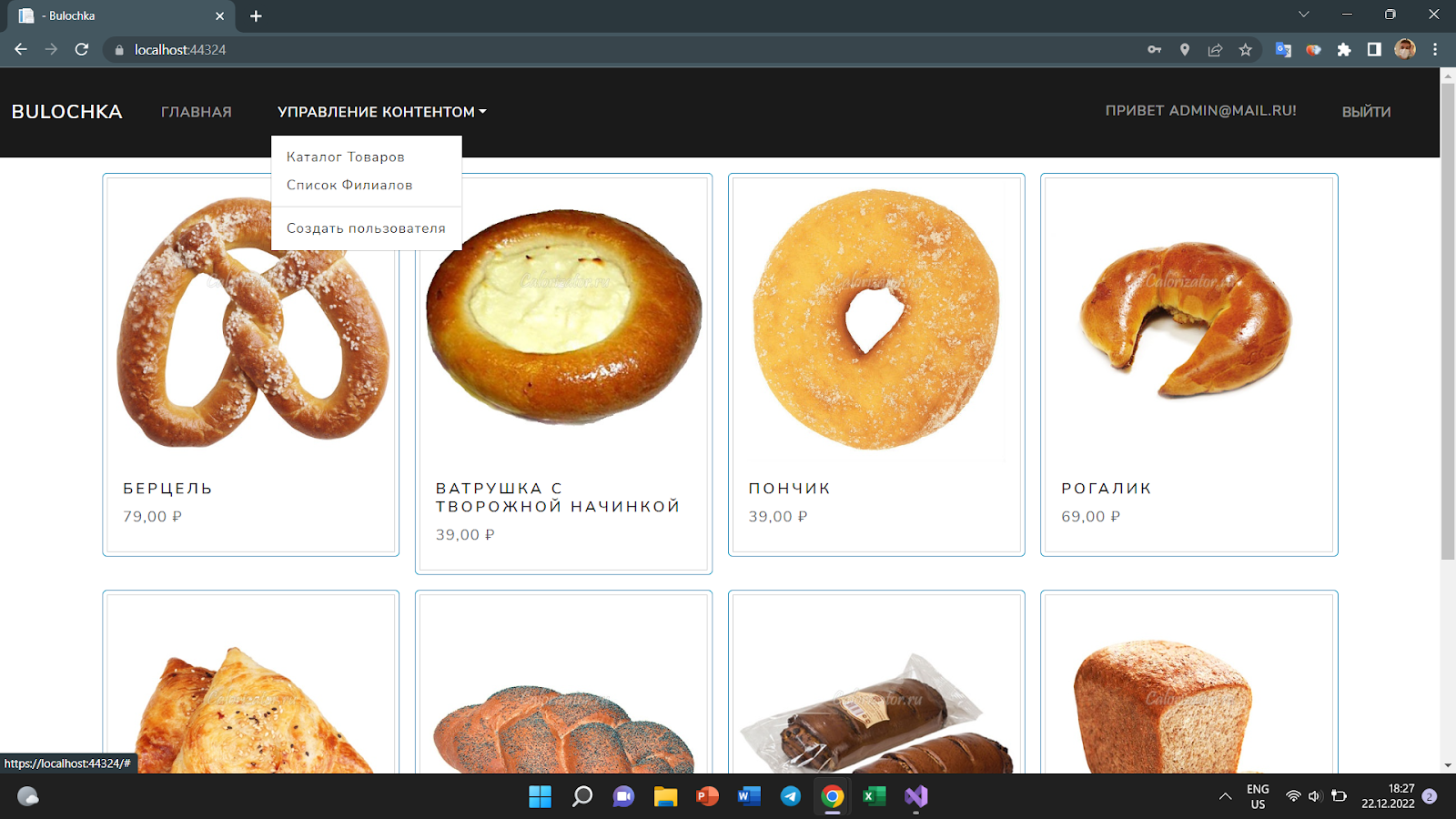


Рисунок 13 – Управление контентом

При регистрации пользователя, если ему выбрана роль «Работник», появляется возможность также выбрать предприятие (филиал), в котором он работает. Также администратор може регистрировать в системе другого администратора. В остальном же регистрация пользователя в системе не отличается от стандартной.

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Рисунок 14 – Регистрация (портал администратора)

Отображение всех объектов реализовано с помощью datatables.net, использование которых позволяет без написание лишнего кода пользоваться сортировкой, поиском элементов и фильтрами отображения таблицы. При открытии страницы каталога вызывается js-функция, которая на основе API запроса генерирует таблицу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15 – Получение всех товаров

Логика добавление товаров проста и понятна, однако с изменением товаров пришлось потратить немного больше времени для более эффективной работы контроллера, так как при изменении картинки товара мы не хотим оставлять старую картинку в папке, потому что она занимает место. Было решено реализовать логику удаления старой картинки (если таковая имеется) и только после этого добавлять новую.

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Рисунок 16 – Обработка файла картинки

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 – Метод удаления старой картинки

При удалении товара всплывает окно дополнительного подтверждения для избежания случайного удаления данных. Если пользователь уверен, то вместе с удалением товара из базы данных, его картинка также удаляется из папки как было описано выше.

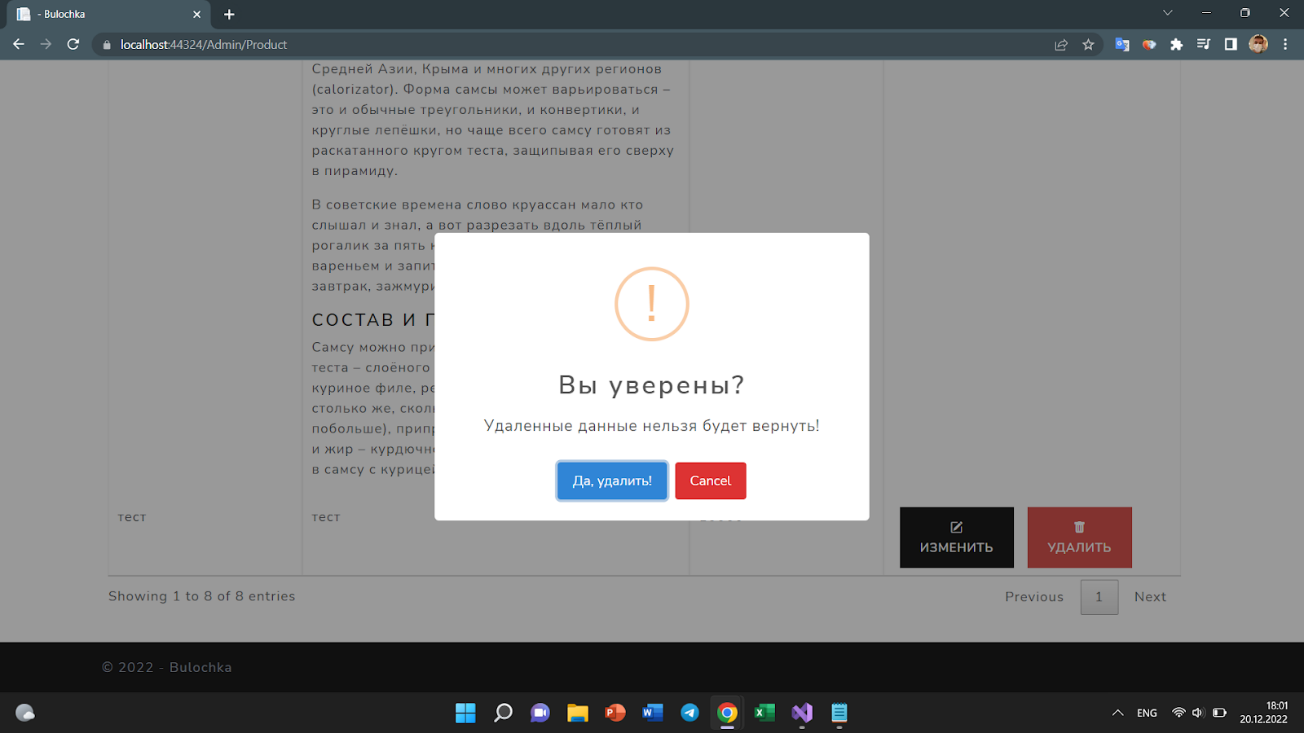


Рисунок 18 – Окно подтверждения удаления

Все функции системы были успешно протестированы. Каталог был заполнен несколькими товарами, список филиалов также был заполнен некоторыми данными. В системы были зарегистрированы пользователи со всеми типами ролей и были проверены их возможности взаимодействия с системой.

**Выводы**

Было создано web-приложение, реализующее запросы для доступа к данным.