**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 курсовому проекту

по дисциплине «Избранные главы информатики»

на тему: **«Сайт для управления складом аптеки»**

Исполнитель: студент гр. ИП-31 Зайцев М. С.

Руководитель: старший преподаватель

Романькова Т. Л.

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

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

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

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

Подписи членов комиссии

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

Гомель 2021

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc72125720)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc72125721)

[1.1 Аналитический обзор существующих аналогов 4](#_Toc72125722)

[1.2 Требования к проектируемому программному обеспечению 5](#_Toc72125723)

[1.3 Обзор используемых средств 6](#_Toc72125724)

[1.4 Выводы аналитического обзора 7](#_Toc72125725)

[2 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 8](#_Toc72125726)

[2.1 Общая информация 8](#_Toc72125727)

[2.3 Прецеденты и их описание 11](#_Toc72125728)

[2.4 Проектирование базы данных 15](#_Toc72125729)

[3 ОПИСАНИЕ АРХИТЕКТУРЫ 18](#_Toc72125730)

[3.1 Описание классов 18](#_Toc72125731)

[3.2 Описание интерфейса 18](#_Toc72125732)

[4 ТЕСТИРОВАНИЕ 27](#_Toc72125733)

[4.1 Обработка исключительных ситуаций 27](#_Toc72125734)

[ЗАКЛЮЧЕНИЕ 30](#_Toc72125735)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc72125736)

# 

# **ВВЕДЕНИЕ**

В современном мире управление складом без применения информационных технологий достаточно трудоемкая и тормозящая процесс обработки данных задача. Возможности автоматизации приобрели комплексный характер, который охватывает все процессы деятельности не только различных организаций, занимающихся оптовой торговлей, но и других организаций, где требуется активное взаимодействие с большим количеством данных.

Компьютерные системы управления позволяют автоматизировать выполнение рутинных задач персонала и руководства. Использование автоматизированных систем делает организацию более управляемой. Руководство получает возможность просматривать актуальные данные, что дает возможность принимать корректные и своевременные решения.

Современные предприятия широко разворачивают компьютеризацию своей деятельности, так как поиск и обработка информации вручную занимает очень большой промежуток времени.

Деятельность современных предприятий активно связана с реализацией различных баз данных. Это обосновано тем, что данный раздел информационных технологий имеет значительную степень применимости в практических ситуациях и довольно просто внедряется в каждом конкретном случае. К тому же в настоящее время существует множество всевозможных реализаций баз данных и систем управления базами данных (СУБД).

СУБД становятся наиболее популярным средством обработки табличной информации. Они являются инструментальным средством проектирования баз данных при обработке больших объемов информации.

Целью данной курсовой работы является разработка сайта, который предоставляет автоматизированную систему учёта склада аптеки.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Аналитический обзор существующих аналогов

Аптека – особая специализированная организация системы здравоохранения, занимающаяся изготовлением, фасовкой, анализом и продажей [лекарственных средств](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0).

Сейчас трудно представить современную жизнь без аптеки и для их эффективной работы используют различные приложения, сайты.

Для создания сайта склада аптеки были рассмотрены уже существующие аналоги: tabletka.by, apteka-ot-sklada. Данные сервисы имеют огромную базу препаратов и уже давно себя зарекомендовали с хорошей стороны.

Аналогом склада аптек является сервис tabletka.by. На сайте присутствует удобное разделение на категории и огромный выбор препаратов. Также, данный сайт имеет возможность посмотреть наличие препарата в различных аптеках и возможность увидеть разницу в цене.

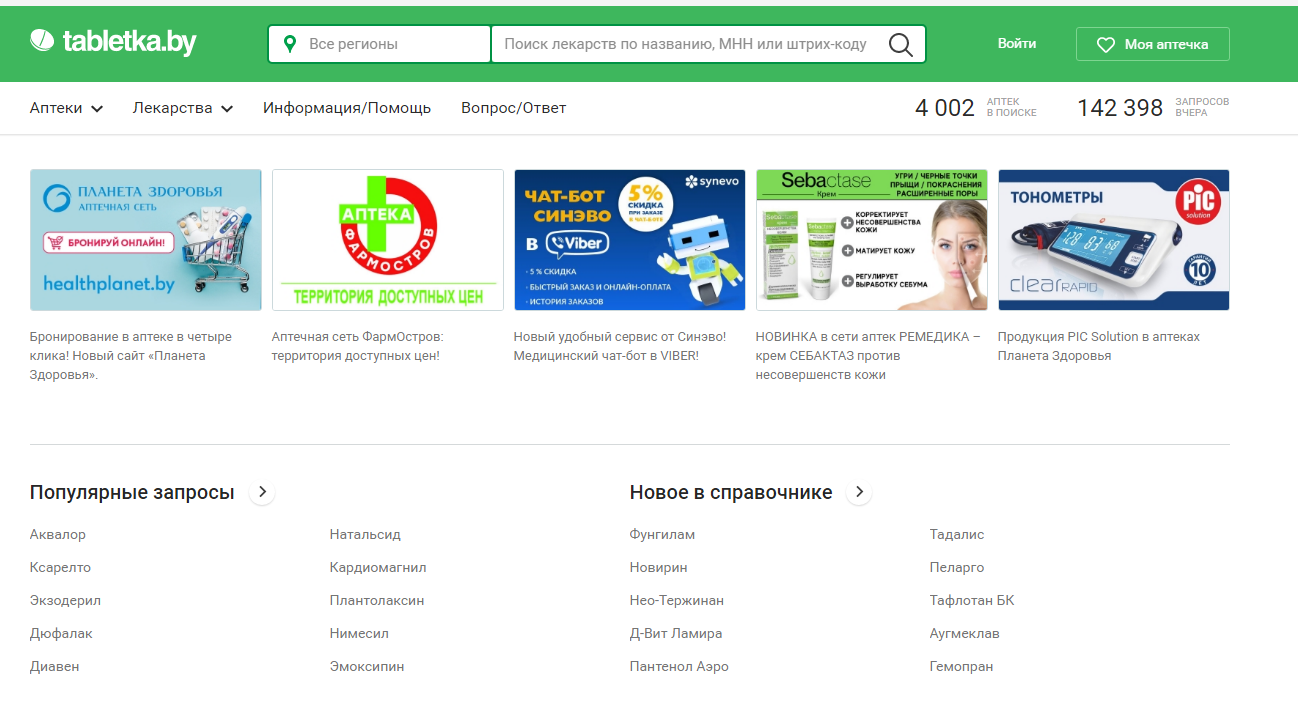


Рисунок 1.1 – Интерфейс сайта «tabletka.by»

Другим аналогам является сайт apteka-ot-sklada. На сайте присутствует все из прошлого аналога, но в нем более понятный интерфейс, больше категорий и больше рекламы, что не очень хорошо. Также присутствует отдельное мобильное приложение.

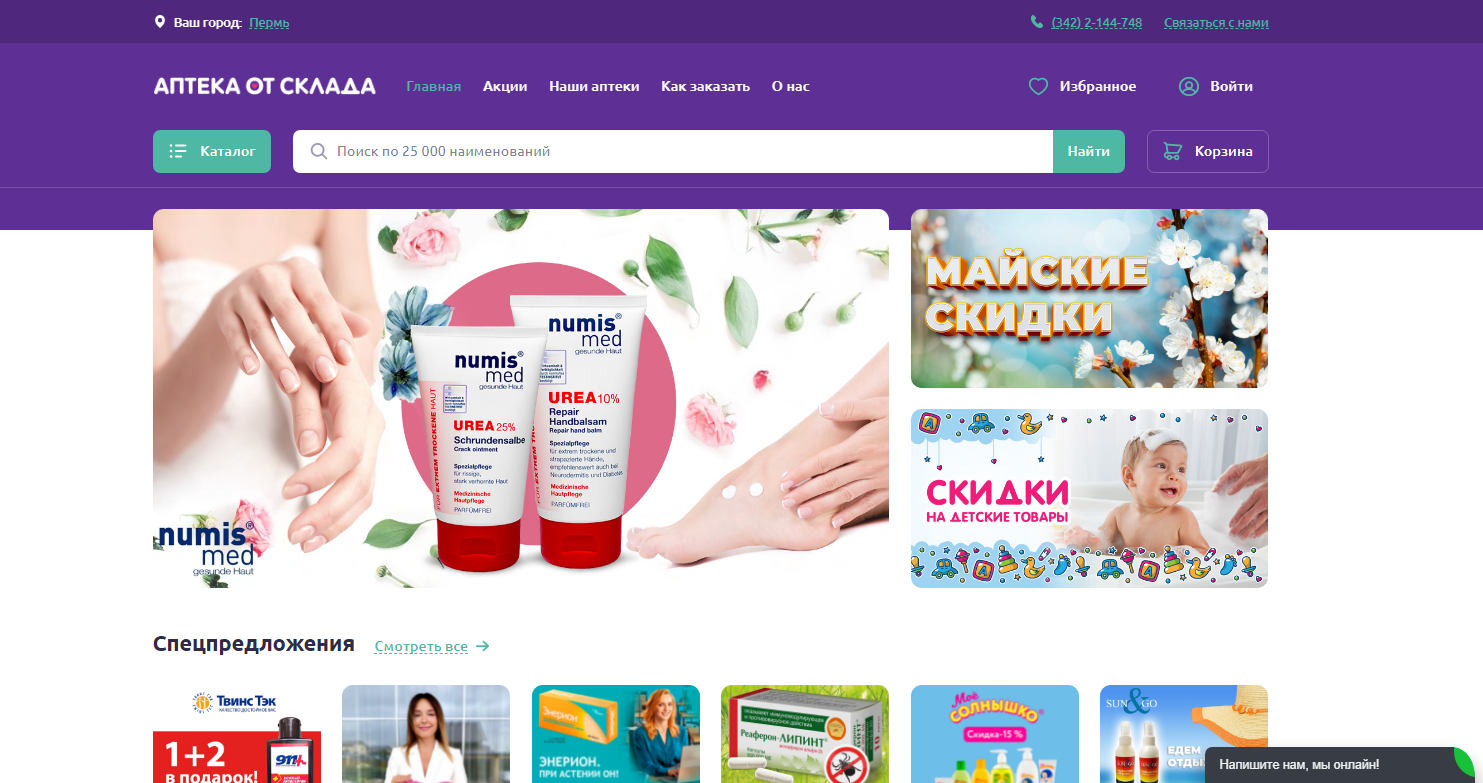


Рисунок 1.2 – Интерфейс сайта «apteka-ot-sklada»

На основе рассмотренных выше средств моделирования была проделана аналитическая работа.

## **1.2 Требования к проектируемому программному обеспечению**

Необходимо разработать web-приложение с использованием технологии ASP.NET MVC, выполняющее следующие функции:

* вход в систему;
* оформление накладных при выдаче товаров;
* списание просроченной продукции;
* получение статистики о движении товаров.

Эти функции работают на основе следующий справочников:

* группы лекарств;
* формы выпуска;
* лекарства.

Приложение должно иметь разделение на роли. Для каждой роли должен быть предусмотрен свой уникальный функционал:

Необходимые роли:

* роль клиента;
* роль гостя;
* роль администратора;
* роль кладовщика.

1.3 Обзор используемых средств

Для создания оконного *Windows*-приложения был выбран объектно-ориентированный язык программирования C#, который является языком разработки приложений для платформы *Microsoft .NET Framework* и основой для использования технелогии *ASP .NET Core*.

Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов.

Для хранения данных используется база данных *MS SQL Server*. Для удобного управления базой данных было использовано специальное средство администрирования SQL Server Management Studio.

Для упрощения работы с базой данных используется *Entity Framework Core –* специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными.

В разробтке приложения использовались такие патерны, как репозиторий, *MVC*.

Паттерн репозиторий является одним из наиболее часто используемых паттернов при работе с данными. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными и остальной программой.

MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

Наиболее качественной средой разработки для языка *C*# на операционной системе *Windows* является *Visual Studio* – официальный продукт от компании *Microsoft* – разработчика платформы *.NET*. Разработка этого проекта велась в среде *Visual Studio 2019 Community.*

1.4 Выводы аналитического обзора

На основе вышеперечисленных пунктов можно сделать вывод о подходе реализации сайта склад аптеки. Для реализации программного обеспечения. Для реализации приложения были использованы следующие паттерны: репозиторий, *MVC*. В качестве среды разработки программного продукта использовался лицензионный программный продукт компании *Microsoft Visual Studio*. Для управления БД использовалась система управления реляционными базами данных *MS SQL Server*.

2 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Общая информация

Разрабатываемое приложение предназначено для работников склада и для их непосредственного начальства. При помощи данного приложения сотрудники могут заполнять данные о поставках на склад новых медикаментов и о списании и редактировать список медикаментов. Начальство имеет возможность контролировать работу починенных. Администратор имеет доступ к базе данных пользователей, что позволяет ему регулировать ее, удаляя неактуальные аккаунты и добавляя новые при необходимости.

2.2 Анализ задания на проектирование

Для реализации склада аптеки были добавлены следующие роли:

* Администратор;
* Клиент;
* Кладовщик;
* Гость.

*2.2.1* Роль «Администратор» представляет собой владельца склада. Для данной роли доступны следующие возможности:

* Добавление сотрудников;
* Увольнение сотрудников;
* Просмотр склада;
* Добавление товара на склад;
* Списание товара;
* Просмотр списанных товаров.

На рисунке 2.1 представлена диаграмма прецедентов роли «Администратор».

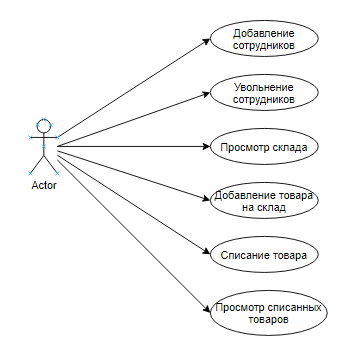


Рисунок 2.1 – Диаграмма прецедентов роли «Администратор»

*2.2.2* Роль «Клиент» является центральной ролью. Для данной роли доступны следующие возможности:

* Просмотр склада;
* Добавить товар в списанные;

На рисунке 2.2 представлена диаграмма прецедентов роли «Клиент».

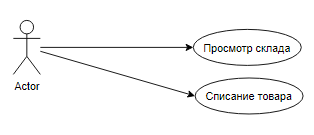


Рисунок 2.2 – Диаграмма прецедентов роли «Клиент»

*2.2.3* Роль «Кладовщик» представляет собой сотрудника склада. Для данной роли доступны следующие роли:

* Просмотр склада;
* Добавление товара на склад;
* Списание товара;
* Просмотр списанных товаров;

На рисунке 2.3 представлена диаграмма прецедентов роли «Кладовщик».



Рисунок 2.3 – Диаграмма прецедентов роли «Кладовщик»

*2.2.3* Роль «Гость» представляет собой незарегистрированного пользователя. Для данной роли доступны следующие роли:

* Просмотр склада;

На рисунке 2.4 представлена диаграмма прецедентов роли «Гость».

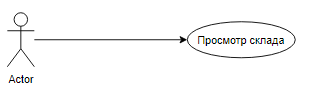


Рисунок 2.4 – Диаграмма прецедентов роли «Гость»

2.3 Прецеденты и их описание

*2.3.1* Преценденты роли «Клиент». Сценарий прецедента «Регистрация»:

* клиент заходит на страницу;
* клиент нажимает кнопку «Зарегистрироваться»;
* клиент вводит данные для регистрации;
* клиент нажимает кнопку «Зарегистрироваться»;
* действие сценария окончено.

Прецендент «Авторизация клиента» представляет себя вход пользователя в свой личный кабинет.

Сценарий прецендента «Авторизация клиента»:

* клиент заходит на страницу;
* клиент вводит данные для авторизации;
* клиент нажимает кнопку «Вход»;
* действие сценария окончено.

Прецендент «Просмотр препаратов на складе» представляет просмотр авторизовавшимся клиентам препаратов на складе.

Сценарий прецендента «Просмотр препаратов на складе»:

* клиент заходит на страницу;
* открывается страница с препаратами на складе;
* действие сценария окончено.

Прецендент «Списание товара со склада» представляет списание авторизовавшимся клиентам препаратов со склада.

Сценарий прецендента «Списание товара со склада»:

* клиент заходит на страницу;
* открывается страница с препаратами на складе;
* клиент выбирает нужный препарат;
* клиент нажимает кнопку «списать»;
* действие сценария окончено.

*2.3.2* Преценденты роли «Администратор». Сценарий прецедента «Авторизация»:

* администратор заходит на страницу;
* администратор вводит данные для авторизации;
* администратор нажимает кнопку «Вход»;
* действие сценария окончено.

Прецендент «Добавление нового сотрудника» представляет добавление нового сотрудника в систему.

Сценарий прецедента «Добавление нового сотрудника»:

* администратор входит на страницу;
* администратор авторизуется;
* администратор нажимает на кнопку «Пользователи»;
* открывается страница с пользователями;
* администратор нажимает на кнопку «Добавить пользователя»;
* администратор вводит все необходимые данные;
* администратор нажимает на кнопку «Добавить»;
* действие сценария окончено.

Прецендент «Увольнение сотрудника» представляет Увольнение сотрудника.

Сценарий прецедента « Увольнение сотрудника»:

* администратор входит на страницу;
* администратор авторизуется;
* администратор нажимает на кнопку «Пользователи»;
* открывается страница с пользователями;
* администратор нажимает на кнопку «Удалить»;
* администратор вводит все необходимые данные;
* администратор нажимает на кнопку «Удалить»;
* действие сценария окончено.

Прецендент «Просмотр препаратов на складе» представляет просмотр администраторам препаратов на складе.

Сценарий прецендента «Просмотр препаратов на складе»:

* администратор заходит на страницу;
* открывается страница с препаратами на складе;
* действие сценария окончено.

Прецендент «Списание товара со склада» представляет списание администраторам препаратов со склада.

Сценарий прецендента «Списание товара со склада»:

* администратор заходит на страницу;
* открывается страница с препаратами на складе;
* администратор выбирает нужный препарат;
* администратор нажимает кнопку «списать»;
* действие сценария окончено.

Прецендент «Добавление товара на склад» представляет добавление администраторам препаратов на склад.

Сценарий прецендента «Добавление товара на склад»:

* администратор заходит на страницу;
* открывается страница с препаратами на складе;
* администратор нажимает кнопку «Добавить»;
* администратор вводит все необходимые данные;
* администратор нажимает кнопку «Добавить»;
* действие сценария окончено.

Прецендент «Просмотр листа ожидания» представляет просмотр администраторам препаратов в листе ожидания.

Сценарий прецендента «Просмотр листа ожидания»:

* администратор заходит на страницу;
* открывается страница с препаратами на складе;
* администратор нажимает кнопку «Список поступивших препаратов»;
* открывается страница с препаратами в листе ожидания;
* действие сценария окончено.

Прецендент «Добавление препарата в лист ожидания» представляет добавление администраторам препаратов в лист ожидания.

Сценарий прецендента «Добавление препарата в лист ожидания»:

* администратор заходит на страницу;
* открывается страница с препаратами на складе;
* администратор нажимает кнопку «Список поступивших препаратов»;
* открывается страница с препаратами в листе ожидания;
* администратор нажимает кнопку «Добавить»;
* администратор вводит все необходимые данные;
* администратор нажимает кнопку «Добавить»;
* действие сценария окончено.

Прецендент «Добавление товара с списанные» представляет добавление администраторам препаратов в лист списанных.

Сценарий прецендента «Добавление товара с списанные»:

* администратор заходит на страницу;
* открывается страница с препаратами на складе;
* администратор выбирает нужный препарат;
* администратор нажимает кнопку «списать»;
* действие сценария окончено.

*2.3.3* Преценденты роли «Кладовщик». Сценарий прецедента «Авторизация»:

* кладовщик заходит на страницу;
* кладовщик вводит данные для авторизации;
* кладовщик нажимает кнопку «Вход»;
* действие сценария окончено.

Прецендент «Просмотр препаратов на складе» представляет просмотр кладовщиком препаратов на складе.

Сценарий прецендента «Просмотр препаратов на складе»:

* кладовщик заходит на страницу;
* открывается страница с препаратами на складе;
* действие сценария окончено.

Прецендент «Списание товара со склада» представляет списание кладовщиком препаратов со склада.

Сценарий прецендента «Списание товара со склада»:

* кладовщик заходит на страницу;
* открывается страница с препаратами на складе;
* кладовщик выбирает нужный препарат;
* кладовщик нажимает кнопку «списать»;
* действие сценария окончено.

Прецендент «Добавление товара на склад» представляет добавление кладовщиком препаратов на склад.

Сценарий прецендента «Добавление товара на склад»:

* кладовщик заходит на страницу;
* открывается страница с препаратами на складе;
* кладовщик нажимает кнопку «Добавить»;
* кладовщик вводит все необходимые данные;
* кладовщик нажимает кнопку «Добавить»;
* действие сценария окончено.

Прецендент «Просмотр листа ожидания» представляет просмотр кладовщиком препаратов в листе ожидания.

Сценарий прецендента «Просмотр листа ожидания»:

* кладовщик заходит на страницу;
* открывается страница с препаратами на складе;
* кладовщик нажимает кнопку «Список поступивших препаратов»;
* открывается страница с препаратами в листе ожидания;
* действие сценария окончено.

Прецендент «Добавление препарата в лист ожидания» представляет добавление кладовщиком препаратов в лист ожидания.

Сценарий прецендента «Добавление препарата в лист ожидания»:

* кладовщик заходит на страницу;
* открывается страница с препаратами на складе;
* кладовщик нажимает кнопку «Список поступивших препаратов»;
* открывается страница с препаратами в листе ожидания;
* кладовщик нажимает кнопку «Добавить»;
* кладовщик вводит все необходимые данные;
* кладовщик нажимает кнопку «Добавить»;
* действие сценария окончено.

Прецендент «Добавление товара с списанные» представляет добавление кладовщиком препаратов в лист списанных.

Сценарий прецендента «Добавление товара с списанные»:

* кладовщик заходит на страницу;
* открывается страница с препаратами на складе;
* кладовщик выбирает нужный препарат;
* кладовщик нажимает кнопку «списать»;
* действие сценария окончено.

2.4 Проектирование базы данных

Для реализации учета склада аптеки в базу данных были добавлены сущности, моделирующие заданную предметную область. В данной модели присутствуют следующие сущности: склад, список ожидания, список списанной продукции, препараты, типы препаратов и форма выпуска препарата.

Обозначим атрибуты для каждой из сущностей в таблицах 2.1-2.6:

Таблица 2.1 – Сущность «Форма выпуска»

|  |  |
| --- | --- |
| Имя | Описание |
| Id | Поле для хранения идентификационного номера формы |
| FormName | Поле для хранения названия формы |

Таблица 2.2 – Сущность «Препараты»

|  |  |
| --- | --- |
| Имя | Описание |
| Id | Поле для хранения идентификационного номера препарата |
| Name | Поле для хранения названия формы |
| MedicalBillsTypeId | Поле для хранения идентификационного типа препарата |
| FormId | Поле для хранения идентификационного номера формы |

Таблица 2.3 – Сущность «Тип препарата»

|  |  |
| --- | --- |
| Имя | Описание |
| Id | Поле для хранения идентификационного номера препарата |
| Type | Поле для хранения названия препарата |

Таблица 2.4 – Сущность «Склад»

|  |  |
| --- | --- |
| Имя | Описание |
| Id | Поле для хранения идентификационного номера склада |
| MedicalBillId | Поле для хранения идентификационного номера препарата |
| DateOfManufacture | Поле для хранения даты производства препарата |
| ShelfLife | Поле для хранения срока годности препарата |

Таблица 2.5 – Сущность «Лист ожидания»

|  |  |
| --- | --- |
| Имя | Описание |
| Id | Поле для хранения идентификационного списка ожидания |
| MedicalBillId | Поле для хранения идентификационного номера препарата |
| DateOfManufacture | Поле для хранения даты производства препарата |
| ShelfLife | Поле для хранения срока годности препарата |

Таблица 2.6 – Сущность «Лист списанных»

|  |  |
| --- | --- |
| Имя | Описание |
| Id | Поле для хранения идентификационного номера списка списанных препаратов |
| MedicalBillId | Поле для хранения идентификационного номера препарата |
| DateOfManufacture | Поле для хранения даты производства препарата |
| ShelfLife | Поле для хранения срока годности препарата |

На рисунке 2.5 изображена физическая модель базы данных

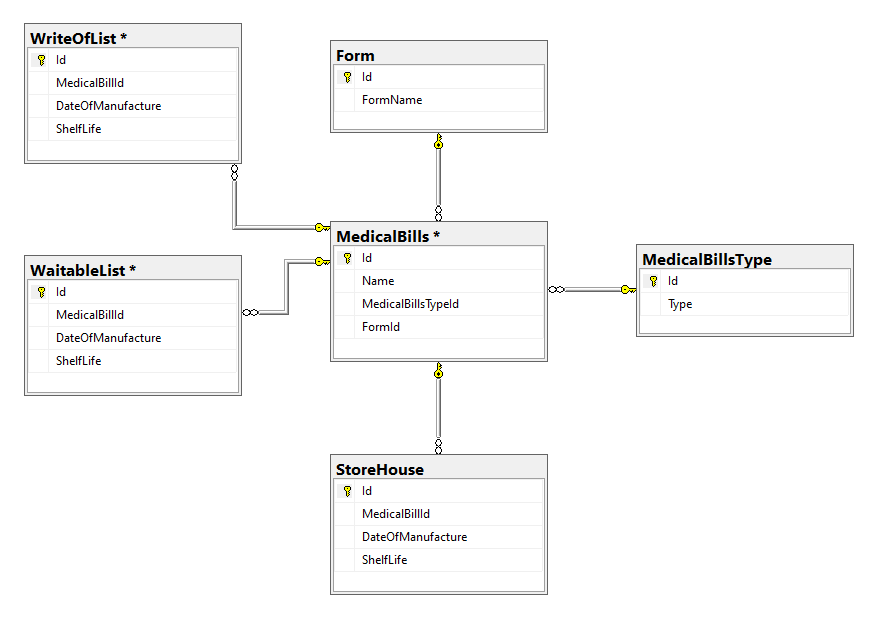


Рисунок 2.5 – Физическая модель базы данных

3 ОПИСАНИЕ АРХИТЕКТУРЫ

3.1 Описание классов

Приложение состоит из четырёх слоёв.

Первый слой представляет собой совокупность классов, представляющих сущности таблицы.

Второй слой представляет логику для работы с источником данных. Для реализации логики для работы с источником данных был реализован паттерн “Репозиторий”. Данный паттерн позволяет разделять слой доступа к данным и безнес-логику.

Третий слой представляет собой бизнес-логику. Он инкапсулирует всю бизнес-логику, все необходимые вычисления, получает объекты из уровня доступа к данным и передает их на уровень представления, либо, наоборот, получает данные с уровня представления и передает их на уровень данных.

Четвёртый слой представляет собой веб-приложение. Веб-приложение отвечает непосредственно за передачу данных на представление для пользователя.

3.2 Описание интерфейса

После запуска программы пользователь увидит следующую страницу:

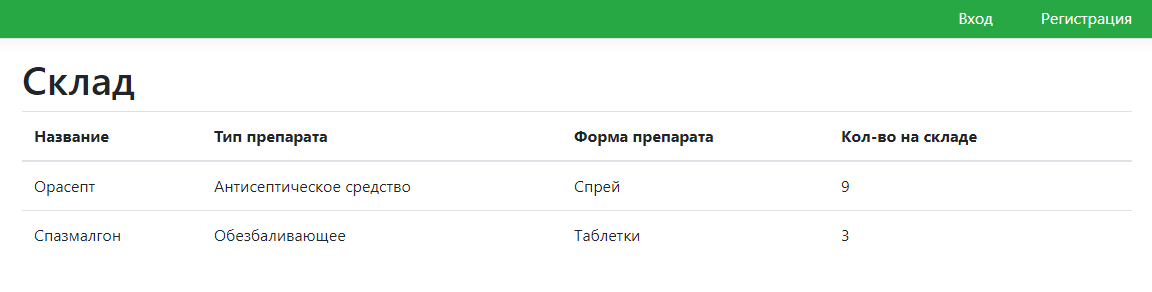


Рисунок 3.1 – Главная страница

В стартовом окне имеется возможность авторизоваться при нажатии на кнопку «Вход». Страница авторизации изображена на рисунке 3.2:

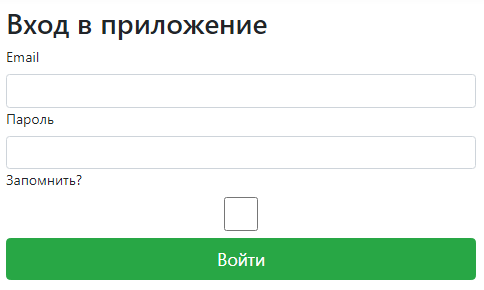


Рисунок 3.2 – Страница авторизации

В стартовом окне имеется возможность зарегистрироваться при нажатии на кнопку «Регистрации». Страница авторизации изображена на рисунке 3.3:

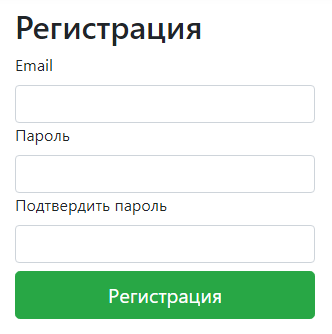


Рисунок 3.3 – Страница авторизации

После авторизации или регистрации стартовая страница изменяется.

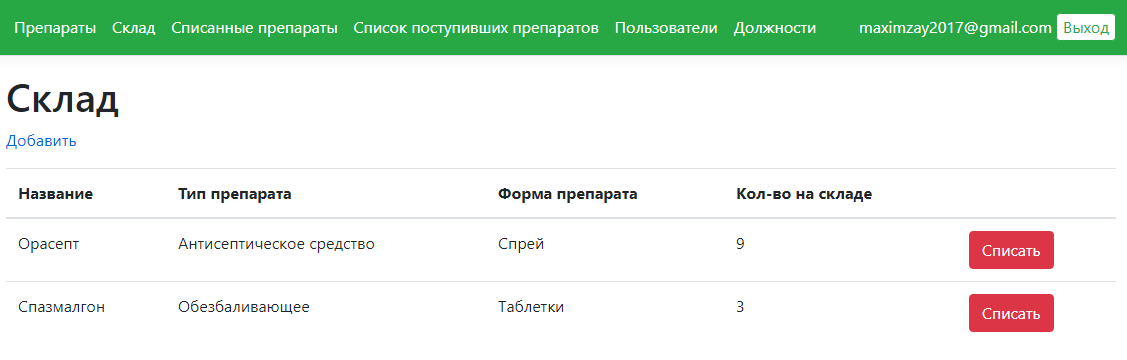


Рисунок 3.4 – Новая стартовая страница

После нажатия на кнопку «Препараты» появится страница со списком препаратов. Страница списка препаратов изображена на рисунке 3.5:

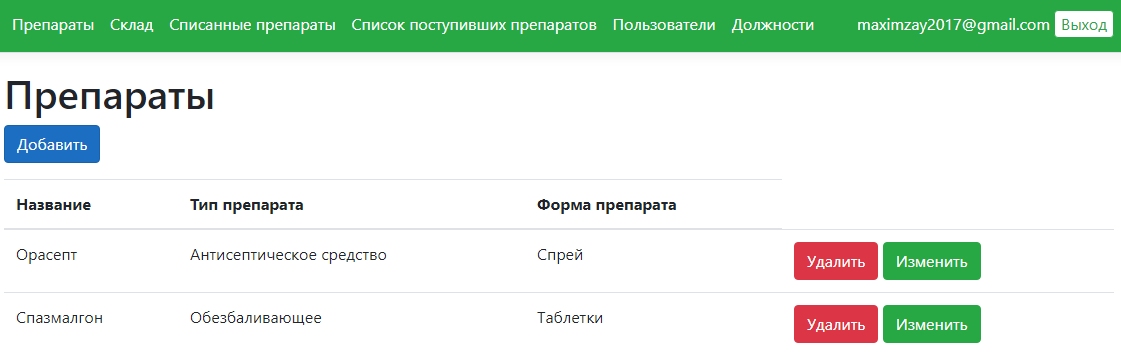


Рисунок 3.5 – Страница со списком препаратов

Администратор и кладовщик имеют возможность добавлять новые препараты.

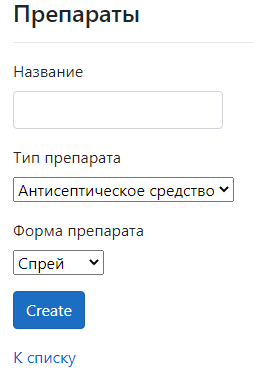
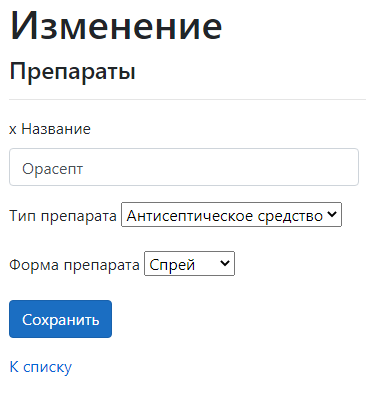


Рисунок 3.5 – Страница добавления препаратов

Администратор и кладовщик имеют возможность изменять уже существующие препараты.



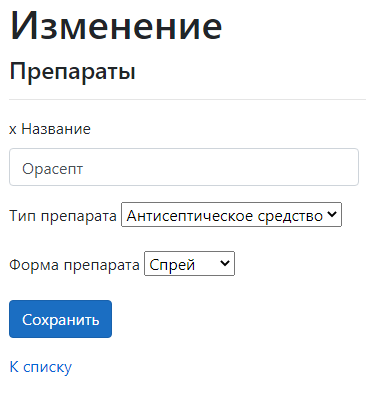


Рисунок 3.6 – Страница изменения препаратов

Администратор и кладовщик имеют возможность добавлять препараты на склад.

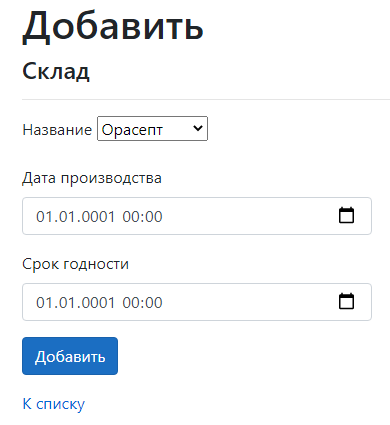


Рисунок 3.7 – Страница добавления препаратов на склад

Администратор и кладовщик имеют возможность просматривать список списанных препаратов.

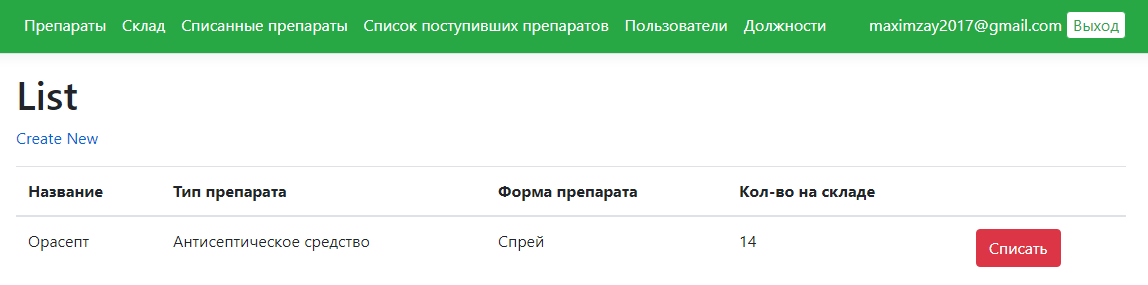


Рисунок 3.8 – Страница просмотра списанных препаратов

Администратор и кладовщик имеют возможность просматривать список препаратов в списке ожидания.

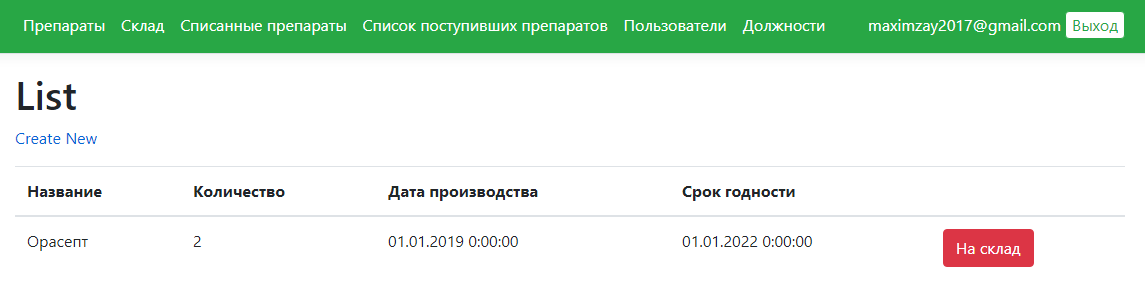


Рисунок 3.9 – Страница просмотра списка ожидания

Администратор и кладовщик имеют возможность добавлять препараты в список ожидания.

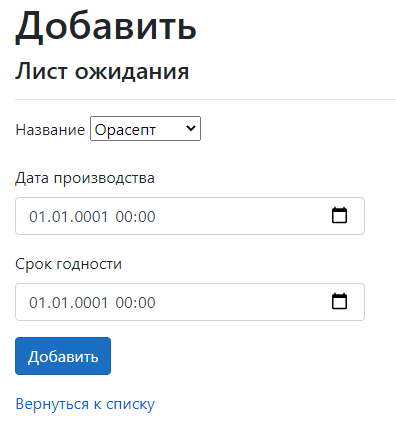


Рисунок 3.10 – Страница добавления препарата к списку ожидания

Администратор имеет возможность просматривать список пользователей

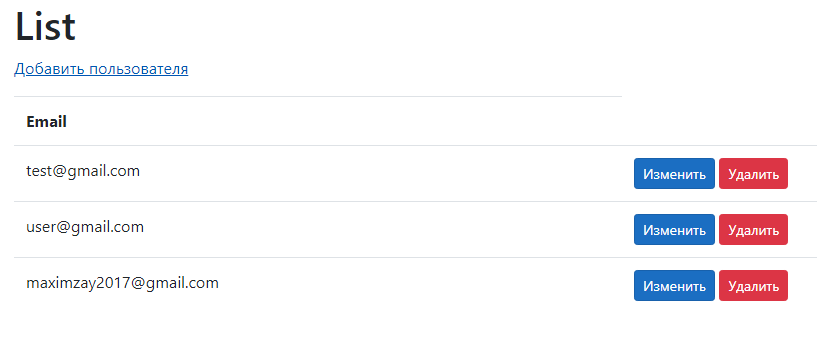


Рисунок 3.11 – Страница управления пользователями

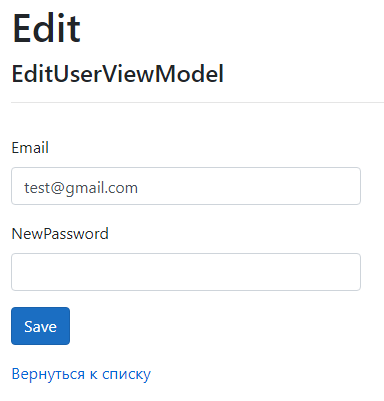


Рисунок 3.12 – Страница изменения пользователя

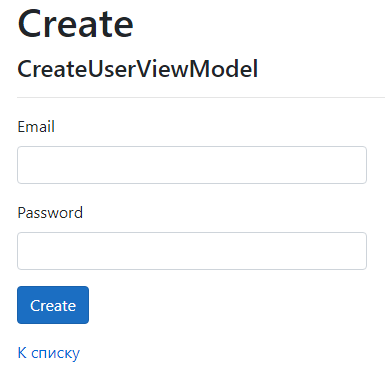


Рисунок 3.13 – Страница добавления пользователя

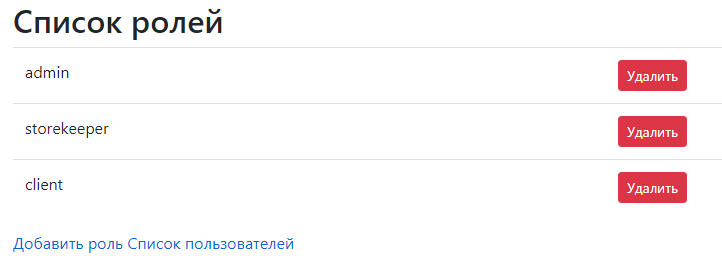


Рисунок 3.14 – Страница управления ролями

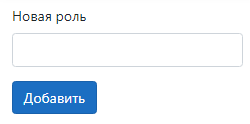


Рисунок 3.14 – Страница добавления роли

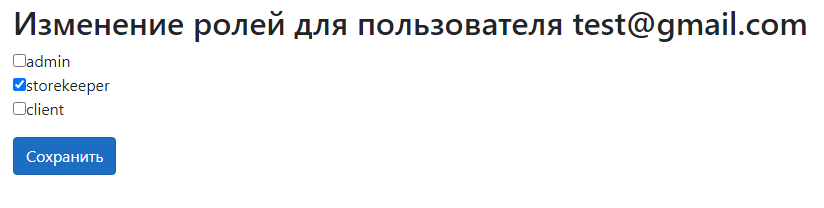


Рисунок 3.15 – Страница изменения роли

4 ТЕСТИРОВАНИЕ

4.1 Обработка исключительных ситуаций

Приложение имеет обработку некорректного ввода данных. В случае исключительной ситуации пользователь получит уведомление об ошибке.

На рисунках 4.1 – 4.4 изображена попытка ввода некорректных данных.

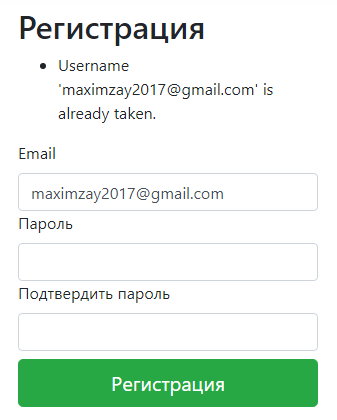


Рисунок 4.1 – Попытка ввода некорректных данных

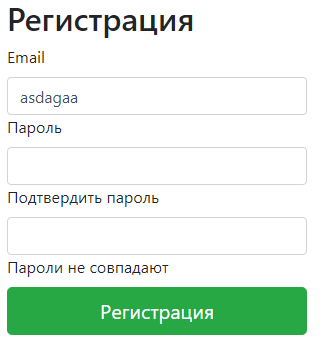


Рисунок 4.2 – Попытка ввода некорректных данных

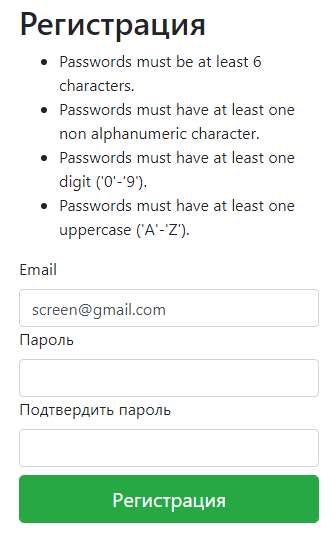


Рисунок 4.3 – Попытка ввода некорректных данных

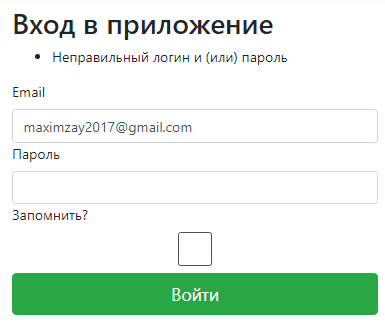


Рисунок 4.4 – Попытка ввода некорректных данных

ЗАКЛЮЧЕНИЕ

В курсовой работе было создано web-приложение, предназначенное для автоматизации управления складом аптеки. Это приложение имеет следующий функционал:

* ведение справочников;
* авторизация и регистрация пользователей;
* хранение данных

В процессе работы были поставлены основные задачи и подготовлены решения для выполнения основных технических требований. Поставленная задача была выполнена при помощи объектно-ориентированного языка *C#*, с использованием технологии *ASP .NET MVC CORE* в среде разработки *MS Visual Studio 2019*. Для хранения данных была использована база данных *MS SQL*.

Данный проект требует значительных доработок, чтобы хоть как-то пытаться конкурировать с аналогами на рынке. Из добавок можно отметить добавление новых функций, добавление мобильной версии, доработать дизайн.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рихтер, Дж. CLR via С#. Программирование на платформе Microsoft .NET Framework 4.0 на языке С#. 3-е изд. - СПб.: Питер, 2012. – 928 с.: ил.

2. Стиллмен, Э., Грин, Дж. Изучаем C#. 3-е изд. – СПб.: Питер, 2014. – 816 с.: ил.

3. Фримен, Адам ASP.NET 4.5 с примерами для профессионалов / Адам Фримен. – Москва: Диалектика / Вильямс, 2014. – 10-23 с.

4.Троелсен, Э. Язык программирования *C#* 5.0 и платформа .NET 4.5, 6-е издание. – М.: Вильямс, 2013. – 1312 с.

## **ПРИЛОЖЕНИЕ A**

**Приложение 1. Листинг программы**

Название: MedicalBillsRepository

using System;

using System.Collections.Generic;

using DAL.Interfaces;

using DAL.Entities;

using DAL.Contexts;

using System.Linq;

using Microsoft.EntityFrameworkCore;

namespace DAL.Repositories

{

public class MedicalBillsRepository : IMedicalBills

{

ApplicationContext db = new ApplicationContext();

public void Add(MedicalBills entity)

{

db.Attach(entity.MedicalBillsType);

db.Attach(entity.Form);

db.MedicalBills.Add(entity);

db.SaveChanges();

}

public IEnumerable<MedicalBills> GetAll()

{

return db.MedicalBills.Include(t => t.MedicalBillsType).Include(u => u.Form).AsNoTracking();

}

public void Remove(MedicalBills entity)

{

db.MedicalBills.Remove(db.MedicalBills.FirstOrDefault(u => u.Id == entity.Id));

db.SaveChanges();

}

public void Update(MedicalBills entity)

{

var bill = db.MedicalBills.FirstOrDefault(u => u.Id == entity.Id);

bill.Name = entity.Name;

bill.MedicalBillsType = entity.MedicalBillsType;

bill.MedicalBillsTypeId = entity.MedicalBillsTypeId;

bill.Form = entity.Form;

bill.FormId = entity.FormId;

db.SaveChanges();

}

}

}

**Приложение 2. Листинг программы**

Название: Configuration

using System;

using System.Collections.Generic;

using System.Text;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.DependencyInjection;

using DAL.Contexts;

using DAL.Interfaces;

using DAL.Repositories;

namespace DAL

{

public static class Configuration

{

public static void ConfigurationDalServices(this IServiceCollection services, string connstring)

{

services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(connstring));

services.AddTransient<IMedicalBills, MedicalBillsRepository>();

services.AddTransient<IMedicalBillsType, MedicalBillsTypeRepository>();

services.AddTransient<IForm, FormRepository>();

services.AddTransient<IStoreHouse, StoreHouseRepository>();

services.AddTransient<IWaitableList, WaitableListRepository>();

services.AddTransient<IWriteOfList, WriteOfListRepository>();

}

}

}

**Приложение 3. Листинг программы**

Название: MedicalBillsService

using System;

using System.Collections.Generic;

using System.Text;

using BLL.Interfaces;

using DAL.Entities;

using BLL.DTOs;

using DAL.Interfaces;

namespace BLL.Services

{

public class MedicalBillsService : IMedicalBillsService

{

private readonly IMedicalBills \_repository;

public MedicalBillsService(IMedicalBills medicalBills)

{

\_repository = medicalBills;

}

public void Add(MedicalBillsDTO entity)

{

\_repository.Add(Mapper.FromDTO(entity));

}

public IEnumerable<MedicalBillsDTO> GetAll()

{

var bills = \_repository.GetAll();

List<MedicalBillsDTO> medicals = new List<MedicalBillsDTO>();

foreach(MedicalBills mb in bills)

{

medicals.Add(Mapper.ToDTO(mb));

}

return medicals;

}

public void Remove(MedicalBillsDTO entity)

{

\_repository.Remove(Mapper.FromDTO(entity));

}

public void Update(MedicalBillsDTO entity)

{

\_repository.Update(Mapper.FromDTO(entity));

}

}

}

**Приложение 4. Листинг программы**

Название: Configuration

using Microsoft.Extensions.DependencyInjection;

using DAL;

using BLL.Interfaces;

using BLL.Services;

namespace BLL

{

public static class Configuration

{

public static void ConfigureBllServices(this IServiceCollection services, string connstring)

{

services.ConfigurationDalServices(connstring);

services.AddTransient<IMedicalBillsService, MedicalBillsService>();

services.AddTransient<IMedicalBillsTypeService, MedicalBillsTypeService>();

services.AddTransient<IFormService, FormService>();

services.AddTransient<IStoreHouseService, StoreHouseService>();

services.AddTransient<IWaitableListService, WaitableListService>();

services.AddTransient<IWriteOfListService, WriteOfListService>();

}

}

}

**Приложение 5. Листинг программы**

Название: Startup

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System.Threading;

using Microsoft.AspNetCore.Http;

using BLL;

using BLL.Interfaces;

using BLL.Services;

using ASP\_UI.Models.Identiy;

using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Identity;

namespace ASP\_UI

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<IdentityContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<User, IdentityRole>()

.AddEntityFrameworkStores<IdentityContext>();

services.ConfigureBllServices(Configuration.GetConnectionString("DefaultConnection"));

// services.AddTransient<IMedicalBillsService, MedicalBillsService>();

services.AddControllersWithViews();

services.AddMvc();

services.AddHttpClient();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseStatusCodePages();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapGet("/Test", async context => {

await context.Response.WriteAsync("<h1>Hello</h1>");

});

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=StoreHouse}/{action=List}/{id?}");

});

}

}

}

**Приложение 6. Листинг программы**

Название: StoreHouseController

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using BLL.DTOs;

using BLL.Interfaces;

using ASP\_UI.Models.ViewModels;

using Microsoft.AspNetCore.Mvc.Rendering;

namespace ASP\_UI.Controllers

{

public class StoreHouseController : Controller

{

private readonly IMedicalBillsService \_medicalBillsService;

private readonly IMedicalBillsTypeService \_medicalBillsTypeService;

private readonly IFormService \_formService;

private readonly IStoreHouseService \_storeHouseServices;

private readonly IWriteOfListService \_writeOfListService;

public IActionResult Index()

{

return View();

}

public StoreHouseController(IMedicalBillsService medicalBillsService, IMedicalBillsTypeService medicalBillsTypeService, IFormService formService, IStoreHouseService storeHouseService, IWriteOfListService writeOfListService)

{

\_medicalBillsService = medicalBillsService;

\_medicalBillsTypeService = medicalBillsTypeService;

\_formService = formService;

\_storeHouseServices = storeHouseService;

\_writeOfListService = writeOfListService;

}

public IActionResult List()

{

List<StoreHouseViewModel> storeViewModel = new List<StoreHouseViewModel>();

var store = \_storeHouseServices.GetAll();

foreach(StoreHouseDTO un in store)

{

if (!storeViewModel.Select(u => u.Name).Contains(un.MedicalBills.Name))

{

StoreHouseViewModel model = new StoreHouseViewModel

{

Id = un.Id,

Name = un.MedicalBills.Name,

Type = un.MedicalBills.MedicalBillsType.Type,

Form = un.MedicalBills.Form.FormName,

Quantity = store.Where(u => u.MedicalBills.Id == un.MedicalBills.Id).Count()

};

storeViewModel.Add(model);

}

}

return View(storeViewModel);

}

public IActionResult Create()

{

var viewModel = new StoreHouseViewModel

{

TypeList = new SelectList(\_medicalBillsTypeService.GetAll().ToList()),

FormList = new SelectList(\_formService.GetAll().ToList()),

MedicalBills = new SelectList(\_medicalBillsService.GetAll())

};

return View(viewModel);

}

[HttpPost]

public IActionResult Create(StoreHouseViewModel storeHouse)

{

if (ModelState.IsValid)

{

var store = new StoreHouseDTO

{

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = \_medicalBillsService.GetAll().FirstOrDefault(u => u.Name == storeHouse.Name)

};

\_storeHouseServices.Add(store);

return RedirectToAction("List");

}

return View();

}

public IActionResult Delete(int id)

{

StoreHouseDTO store = \_storeHouseServices.GetAll().FirstOrDefault(u => u.Id == id);

StoreHouseViewModel modelViewModel = new StoreHouseViewModel

{

Id = store.Id,

Name = store.MedicalBills.Name,

Type = store.MedicalBills.MedicalBillsType.Type,

Form = store.MedicalBills.Form.FormName,

Quantity = 1

};

return View(modelViewModel);

}

[HttpPost]

public IActionResult Delete(StoreHouseViewModel storeHouseViewModel)

{

try

{

List<StoreHouseDTO> store = \_storeHouseServices.GetAll().ToList();

for (int i = 0; i < storeHouseViewModel.Quantity; i++)

{

\_writeOfListService.Add(ConvertTo(store[i]));

}

return RedirectToAction("List");

}

catch

{

return View();

}

}

private WriteOfListDTO ConvertTo(StoreHouseDTO storeHouseDTO)

{

WriteOfListDTO writeOfList = new WriteOfListDTO

{

MedicalBills = storeHouseDTO.MedicalBills,

DateOfManufacture = storeHouseDTO.DateOfManufacture,

ShelfLife = storeHouseDTO.ShelfLife

};

return writeOfList;

}

}

}

**Приложение 6. Листинг программы**

Название: MedicalBillsDTO

using System;

using System.Collections.Generic;

using System.Text;

namespace BLL.DTOs

{

public class MedicalBillsDTO

{

public int Id { get; set; }

public string Name { get; set; }

public MedicalBillsTypeDTO MedicalBillsType { get; set; }

public FormDTO Form { get; set; }

public override string ToString()

{

return Name;

}

}

}

**Приложение 7. Листинг программы**

Название: Mapper

using System;

using System.Collections.Generic;

using System.Text;

using BLL.DTOs;

using DAL.Entities;

namespace BLL.DTOs

{

public static class Mapper

{

public static MedicalBillsDTO ToDTO(MedicalBills medicalBills)

{

MedicalBillsDTO medical = new MedicalBillsDTO

{

Id = medicalBills.Id,

Name = medicalBills.Name,

MedicalBillsType = ToDTO(medicalBills.MedicalBillsType),

Form = ToDTO(medicalBills.Form)

};

return medical;

}

public static MedicalBills FromDTO(MedicalBillsDTO medicalBillsDTO)

{

MedicalBills medicalBills = new MedicalBills

{

Id = medicalBillsDTO.Id,

Name = medicalBillsDTO.Name,

MedicalBillsType = FromDTO(medicalBillsDTO.MedicalBillsType),

Form = FromDTO(medicalBillsDTO.Form)

};

return medicalBills;

}

public static MedicalBillsTypeDTO ToDTO(MedicalBillsType medicalBillsType)

{

MedicalBillsTypeDTO medicalBillsTypeDTO = new MedicalBillsTypeDTO

{

Id = medicalBillsType.Id,

Type = medicalBillsType.Type

};

return medicalBillsTypeDTO;

}

public static MedicalBillsType FromDTO(MedicalBillsTypeDTO medicalBillsTypeDTO)

{

MedicalBillsType medicalBillsType = new MedicalBillsType

{

Id = medicalBillsTypeDTO.Id,

Type = medicalBillsTypeDTO.Type

};

return medicalBillsType;

}

public static FormDTO ToDTO(Form form)

{

FormDTO f = new FormDTO

{

Id = form.Id,

FormName = form.FormName

};

return f;

}

public static Form FromDTO(FormDTO form)

{

Form f = new Form

{

Id = form.Id,

FormName = form.FormName

};

return f;

}

public static StoreHouseDTO ToDTO(StoreHouse storeHouse)

{

StoreHouseDTO f = new StoreHouseDTO

{

Id = storeHouse.Id,

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = ToDTO(storeHouse.MedicalBills)

};

return f;

}

public static StoreHouse FromDTO(StoreHouseDTO storeHouse)

{

StoreHouse f = new StoreHouse

{

Id = storeHouse.Id,

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = FromDTO(storeHouse.MedicalBills)

};

return f;

}

public static WaitableListDTO ToDTO(WaitableList storeHouse)

{

WaitableListDTO f = new WaitableListDTO

{

Id = storeHouse.Id,

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = ToDTO(storeHouse.MedicalBills)

};

return f;

}

public static WaitableList FromDTO(WaitableListDTO storeHouse)

{

WaitableList f = new WaitableList

{

Id = storeHouse.Id,

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = FromDTO(storeHouse.MedicalBills)

};

return f;

}

public static WriteOfListDTO ToDTO(WriteOfList storeHouse)

{

WriteOfListDTO f = new WriteOfListDTO

{

Id = storeHouse.Id,

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = ToDTO(storeHouse.MedicalBills)

};

return f;

}

public static WriteOfList FromDTO(WriteOfListDTO storeHouse)

{

WriteOfList f = new WriteOfList

{

Id = storeHouse.Id,

DateOfManufacture = storeHouse.DateOfManufacture,

ShelfLife = storeHouse.ShelfLife,

MedicalBills = FromDTO(storeHouse.MedicalBills)

};

return f;

}

}

}