

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий
Кафедра программной инженерии
Специальность 6-05-0612-01 Программная инженерия
Направление специальности 6-05-0612-01 Программная инженерия
(программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»
Тема Программное средство «Медицинский центр»

Исполнитель
студент (ка) 2 курса группы 10 Рублевская Маргарита Владимировна
(Ф.И.О.)
Руководитель работы асс., Мушук А.Н.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой _____
Председатель Мушук А.Н.
(подпись)

Оглавление

Введение.....	5
1. Постановка задачи и обзор литературы.....	6
1.1. Описание функциональности программного средства	6
1.2. Анализ прототипов	6
1.2.1. Сайт «Эксана»	6
1.2.2. Сайт «Клиника Каскад»	7
1.2.3. Сайт «ЛОДЭ»	8
1.2.4. Вывод	10
2. Проектирование архитектуры проекта	11
2.1. Проектирование архитектуры проекта	11
3. Разработка функциональной модели и модели данных ПС	12
3.1. Спецификация функциональных требований.....	12
3.2. Обобщенная структура	13
3.3. Основные классы.....	14
3.4. Основные методы.....	16
3.5. Модель базы данных.....	16
4. Тестирование	18
4.1. Руководство по использованию.....	22
4.1.1. Руководство по использованию администратором.....	23
4.1.2. Руководство по использованию пользователем	25
4.1.3. Руководство по использованию врачом	27
Заключение	29
Список использованных источников	30
Приложение А	31
Приложение Б.....	42
Приложение В.....	43

Введение

В современном обществе многие стремятся вести здоровый образ жизни и думать о своем здоровье, хотя не всегда удается выбрать квалифицированных специалистов и правильные направления лечения. При этом, многие сталкиваются с проблемой сомнения к врачам или недостатком информации о процедурах из-за отсутствия подробных описаний.

Это программное решение станет незаменимым инструментом для самостоятельного бронирования и выбора медицинских процедур с подробными описаниями, не выходя из дома и без необходимости общения с людьми.

Пользователь может легко создать новую учетную запись или войти в существующую через интуитивно понятный интерфейс. Это позволяет ему ознакомиться со всеми врачами медицинского центра и доступными процедурами, а также забронировать нужное время и дату для визита.

Авторизованные пользователи могут просматривать свою личную страницу и редактировать информацию, указанную при регистрации. Кроме того, они могут просматривать назначенные рецепты и отслеживать статус своих заявок.

Это программное решение также полезно для администраторов, которые могут управлять базой данных через административный интерфейс, включая добавление новых врачей и процедур.

Главная цель данного курсового проекта – разработка программного обеспечения, которое реализует все перечисленные функции и решает поставленные задачи.

1. Постановка задачи и обзор литературы

1.1. Описание функциональности программного средства

Программное средство предназначено для оптимизации взаимодействия между администраторами, врачами и пациентами в медицинском учреждении. Оно должно обеспечивать удобный и эффективный доступ к необходимым функциям для каждой категории пользователей, а также способствовать улучшению качества предоставляемых медицинских услуг.

Администраторы имеют возможность управлять ключевыми аспектами работы приложения, включая:

- добавление и удаление врачей;
- управление процедурами;
- добавление талонов для врачей.

Врачи получают доступ к функционалу, который позволяет им эффективно выполнять свои обязанности:

- назначение рецептов;
- обработка заявок.

Пациенты могут пользоваться приложением для упрощения взаимодействия с медицинским учреждением:

- заполнение формы заявки;
- просмотр информации о статусе заявки;
- просмотр врачей и процедур;
- выполнение поисковых запросов, фильтрации;
- управление своей учетной записью;
- редактирование учётной записи;
- написание отзывов.

1.2. Анализ прототипов

Одним из первых этапов в создании программного продукта является анализ прототипов и литературных источников. При изучении сайтов-прототипов было выявлено несколько экземпляров схожих по функциональности. Результат анализа нескольких из них представлен ниже.

1.2.1. Сайт «Эксана»

Первым, в качестве аналога, был выбран медицинский центр «Эксана» [1]. Многопрофильный медицинский центр, предлагающий широкий спектр услуг, включая консультации специалистов, диагностику, лабораторные исследования, физиотерапию и др. Акцент на современные методы диагностики и лечения. Главная страница сайта, представлена на рисунке 1.1.

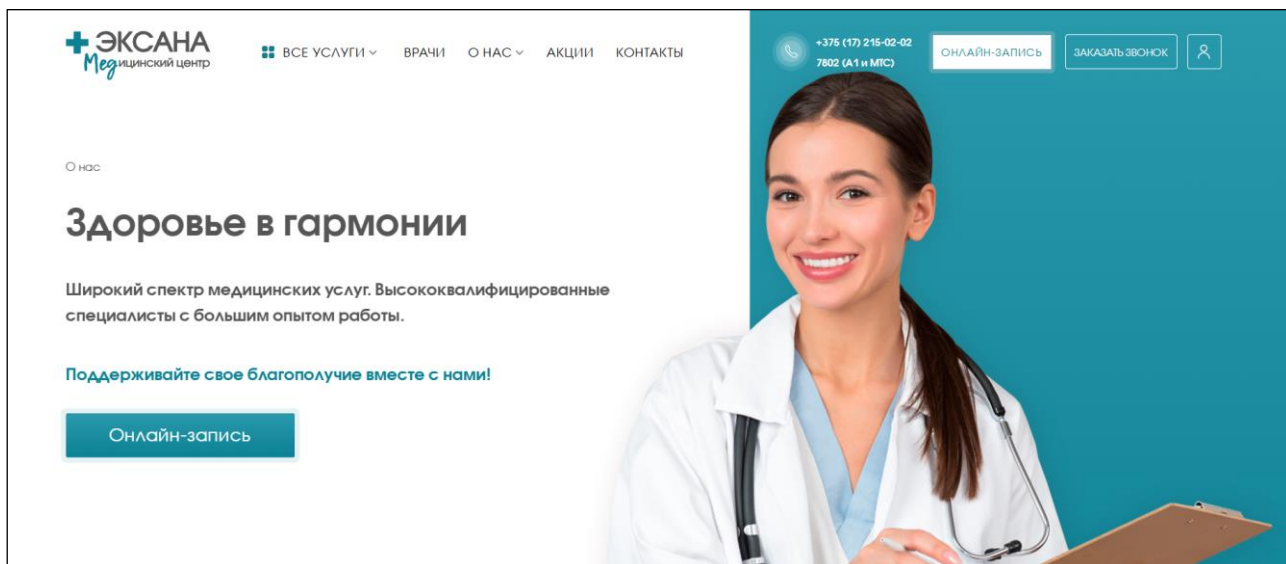


Рисунок 1.1 – Главная страница сайта «Эксана»

Функционал сайта включает базовые возможности:

- 1) Онлайн-запись к врачу.
- 2) Просмотр перечня услуг и специалистов.
- 3) Краткая информация о врачах и направлениях.
- 4) Возможность выбора определенного врача.
- 5) Сортировка по выбору специалиста.

Плюсы сайта:

- 1) Удобный и лаконичный интерфейс, что облегчает навигацию.
- 2) Наличие онлайн-записи к врачу, что экономит время пользователей.
- 3) Возможность просмотра перечня услуг и специалистов, что позволяет пользователям быстро находить необходимую информацию.
- 4) Сортировка по выбору специалиста, что упрощает поиск нужного врача.

Минусы сайта:

- 1) Минимальное количество информации о врачах, что затрудняет процесс выбора подходящего специалиста.
- 2) Недостаток детальной информации о предоставляемых услугах и процедурах, что может вызывать неопределенность у пользователей.
- 3) Ограниченная информативность сайта может снизить доверие пользователей к клинике.

1.2.2. Сайт «Клиника Каскад»

В качестве следующего аналога был выбран медицинский центр «Клиника Каскад» [2]. Медицинский центр, специализирующийся в различных направлениях, включая косметологию, гинекологию, урологию и др. Акцент на современные технологии и индивидуальный подход к пациентам. Главная страница сайта, представлена на рисунке 2.

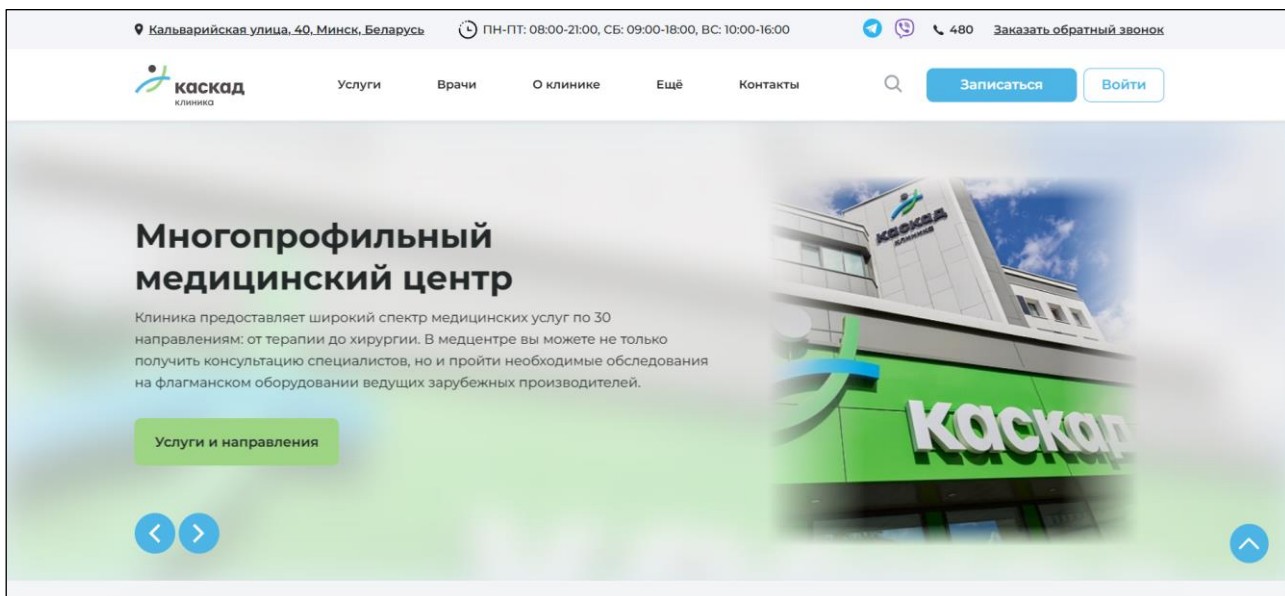


Рисунок 1.2 – Главная страница сайта «Клиника Каскад»

Функционал сайта включает базовые возможности:

- 1) Онлайн-запись к врачу.
- 2) Просмотр перечня услуг и специалистов.
- 3) Краткая информация о врачах и направлениях.
- 4) Возможность выбора определенного врача.
- 5) Сортировка по выбору специалиста.
- 6) Минимальная информация о специалистах.

Плюсы сайта:

- 1) Удобный интерфейс, позволяющий ориентироваться на сайте.
- 2) Наличие онлайн-записи к врачу, что удобно для пользователей.
- 3) Возможность просмотра перечня услуг и специалистов.
- 4) Сортировка по выбору специалиста, что упрощает поиск нужного врача.

Минусы сайта:

- 1) Минимальная информация о специалистах, что может затруднить выбор врача.
- 2) Ограниченная информативность сайта, что делает его менее привлекательным для пользователей.
- 3) Отсутствие разделов с подробными биографиями врачей, описанием услуг и отзывами пациентов, что могло бы повысить доверие и удовлетворенность пользователей.

1.2.3. Сайт «ЛОДЭ»

Третьим аналогом был выбран сайт «ЛОДЭ» – один из крупнейших частных медицинских центров в Беларуси [3].

Медицинский центр «ЛОДЭ» предоставляет широкий спектр медицинских услуг, включая консультации, лабораторные исследования,

диагностику, хирургическое лечение, стоматологию и педиатрию. Сеть имеет множество филиалов по всей стране, включая Минск, Гомель, Брест и др. Главная страница сайта представлена на рисунке 1.3.

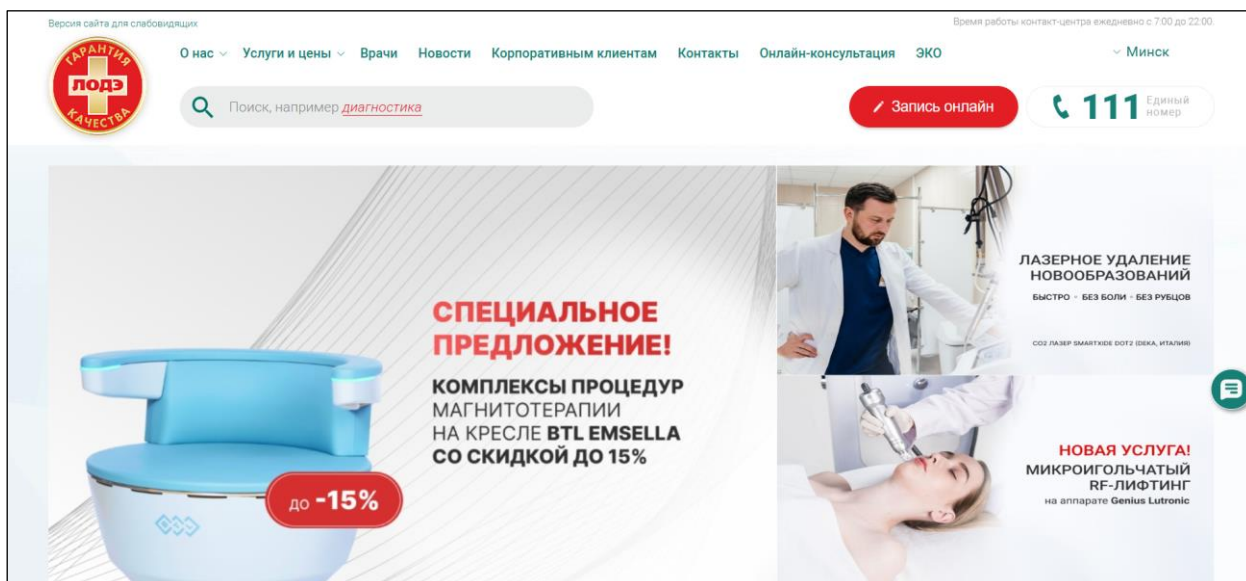


Рисунок 1.3 – Главная страница сайта «ЛОДЭ»

Функционал сайта включает:

- 1) Онлайн-запись к врачу по филиалам;
- 2) Расширенный поиск врача по фамилии, направлению, филиалу;
- 3) Просмотр расписания приёма каждого специалиста;
- 4) Полные биографии врачей с фото, образованием, стажем и специализацией;
- 5) Возможность оформления вызова врача на дом;
- 6) Наличие личного кабинета пациента;
- 7) Поддержка мобильной версии и приложения.

Плюсы сайта:

- 1) Высокий уровень информативности: подробные профили врачей, описание процедур, цены;
- 2) Интерактивность: фильтры, чат-бот, FAQ, быстрая запись;
- 3) Удобство: мобильная версия, приложение, наличие личного кабинета;
- 4) Обратная связь: можно оставить отзыв, воспользоваться формой обратной связи или получить помощь через онлайн-чат.

Минусы сайта:

- 1) Перегруженность интерфейса: большое количество блоков на главной странице может затруднять навигацию;
- 2) Нет единой системы уведомлений (например, напоминаний о приёме на e-mail или по SMS);

3) Ограниченность функционала для незарегистрированных пользователей: без аккаунта нельзя просматривать историю записей или использовать расширенные функции.

1.2.4. Вывод

Анализ вышеперечисленных сайтов медицинских центров позволил выделить как общие черты, так и индивидуальные особенности.

Общие достоинства:

- наличие онлайн-записи;
- просмотр услуг и врачей;
- упрощённый поиск по направлениям;
- чёткое позиционирование услуг.

Общие недостатки:

- недостаток персонализированного опыта (отсутствие личного кабинета на некоторых ресурсах);
- ограниченная информация о врачах;
- нет возможности оценки качества (отзывы, рейтинги);
- слабая визуальная адаптация на мобильных устройствах на некоторых сайтах.

Сайт «ЛОДЭ» продемонстрировал наиболее высокий уровень реализации — личный кабинет, биографии врачей, обратная связь и мобильное приложение. Однако, несмотря на функциональность, он страдает от перегруженности и требует регистрации для доступа ко многим функциям.

На основе анализа можно выделить ключевые рекомендации для собственного проекта: внедрение системы личных кабинетов, подробное описание специалистов и процедур, возможность обратной связи и отзывов.

2. Проектирование архитектуры проекта

Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

2.1. Проектирование архитектуры проекта

На рисунке 2.1 изображена диаграмма вариантов использования данного приложения.

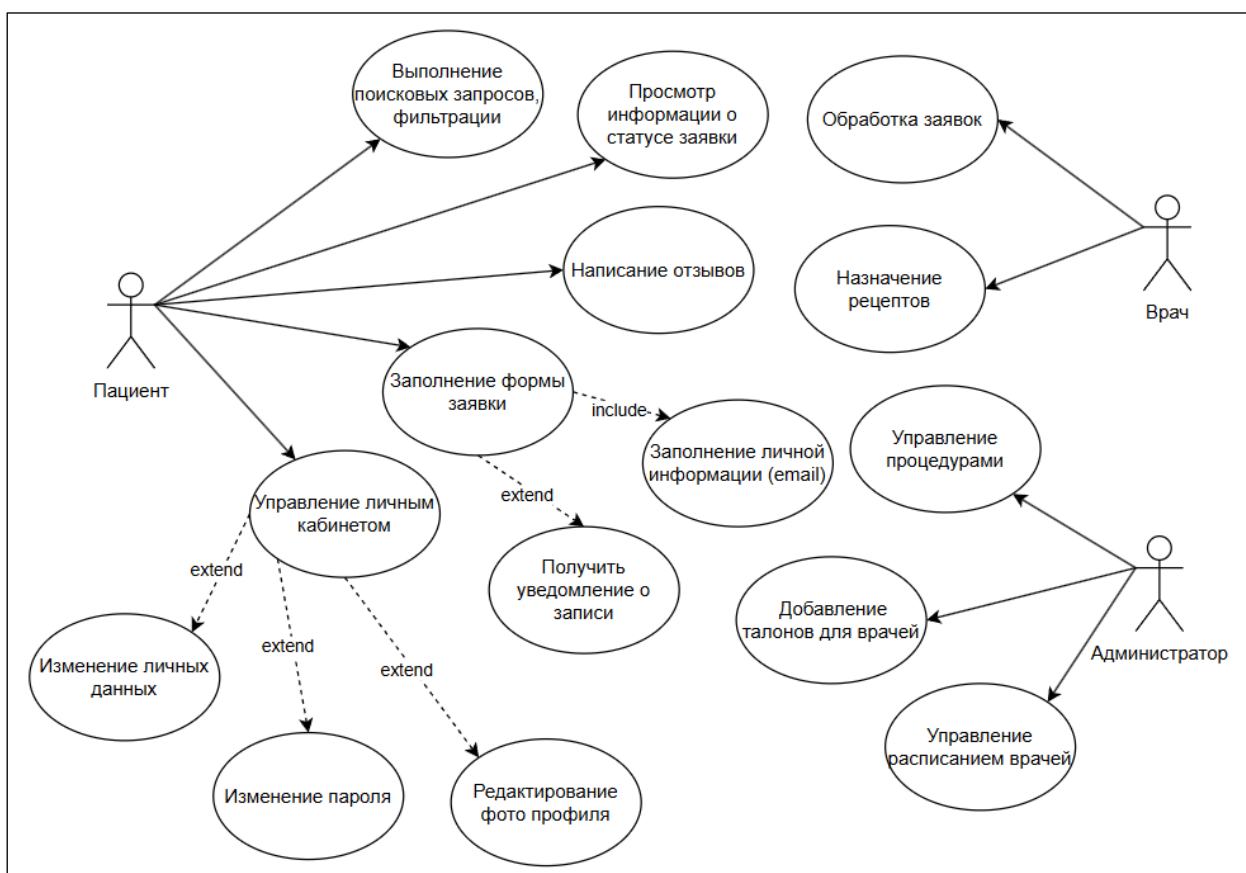


Рисунок 2.1 – Диаграмма вариантов использования для пользователя, администратора и врача

В Приложении В была спроектирована диаграмма последовательности для авторизации. На диаграмме последовательности отображаются только те объекты, которые непосредственно принимают участие во взаимодействии.

В Приложении А представлен программный код, который реализует алгоритм авторизации и регистрации.

3. Разработка функциональной модели и модели данных ПС

3.1. Спецификация функциональных требований

В программном средстве «MedicalCenter» при запуске необходимо реализовать регистрацию или авторизацию для дальнейшего полного использования приложения. Для авторизации пользователю потребуется ввести уникальный логин и пароль, которые должны храниться в базе данных. При регистрации необходимо будет указать имя пользователя, фамилию, электронную почту для получения уведомлений, логин и пароль. Введенные данные должны проходить валидацию: имя и фамилия должны состоять только из кириллицы, логин должен быть уникальным, а почта должна соответствовать общепринятому формату. После успешной валидации данные заносятся в базу данных, и пользователю выдается сообщение о том, что он успешно зарегистрирован и получил полный доступ к приложению.

После входа в свою учетную запись пользователь должен иметь возможность изменить информацию о себе, пароль и изображение, а также просмотреть активные заказы, нажав кнопку «мои заказы» и рецепты, выписанные врачом.

Администратор должен иметь возможность входить в систему через окно логирования, после чего ему будет предоставлен особый функционал. На главной странице администратор сможет кликнуть по кнопке «Посмотреть заказы», что перенесет его на страницу с информацией о каждом заказе. На странице с информацией о врачах администратор сможет добавлять врачей, нажав кнопку в виде плюса, что вызовет дополнительное окно для ввода данных о враче: имя, фамилия, отчество, специальность, логин, пароль, учебная степень, год начала карьеры, изображение и расписание. Если изображение не выбрано, то оно будет вставлено по умолчанию. Здесь же администратор сможет добавлять талоны для определенных врачей, кликнув по кнопке «добавить талоны», что вызовет новое окно. В этом же окне должна быть кнопка «удалить врача», которая откроет новое окно с информацией о врачах и функцией удаления конкретного врача.

На странице о процедурах администратор сможет добавлять процедуры, кликнув по плюсику, что вызовет новое окно-форму для регистрации процедуры. Также он сможет изменять или удалять процедуру, кликнув по кнопке «удалить/изменить».

Врач, заходя через окно логирования, сможет кликнуть по кнопке «Посмотреть заказы», что перенесет его на страницу с информацией о текущих заказах и статусах их выполнения. Кроме того, он сможет воспользоваться кнопками «Подтвердить услугу» и «Назначить рецепт». При нажатии на кнопку «Подтвердить услугу» откроется новое окно, в котором врач сможет подтвердить выполнение услуги для пациента, указав необходимые детали и комментарии. Кнопка «Назначить рецепт» также

откроет форму, где врач сможет ввести информацию о назначении.

Кнопка для выхода из аккаунта должна возвращать пользователя к окну авторизации. Все данные должны заноситься в базу данных в соответствующие таблицы.

3.2. Обобщенная структура

Описание структуры основных файлов представлено в таблице 3.1.
Таблица 3.1 – Описание структуры проекта

Имя файла	Содержание
App.config	Файл с параметрами проекта.
Папка image	Папка, содержащая все изображения.
Properties	Свойства проекта, содержит всю информацию о сборке, используемых ресурсах и настройках.
AddTalon.xaml	Окно для добавления талонов врачам.
RegistrationofDoctor.xaml	Форма для регистрации врачей.
RegistrationofProcedures.xaml	Форма для регистрации процедур.
AddPrescription.xaml	Форма для добавления рецептов
ConfirmService.xaml	Форма подтверждения оказанных услуг
AddReviews.xaml	Форма для создания отзыва.
Createorder.xaml	Форма для создания заказа.
Login.xaml	Окно авторизации.
Profil.xaml	Домашняя страница пользователя.
Registration.xaml	Форма регистрации.
Yourorder.xaml	Окно заказов клиента.
Aboutas.xaml	Окно с информацией о медицинском центре.
allDoctor.xaml	Окно с информацией о врачах.
allprocedures.xaml	Окно с информацией о процедурах.
Main.xaml	Окно-приветствие.
MainHarmony.xaml	Основная форма приложения.
MessangeBox_Ok.xaml	Окно-оповещение.
Reviews.xaml	Окно с отзывами.
App.xaml	Главный класс приложения.
AllOrder.xaml	Окно с информацией о всех заказах.
DeleteDoctor.xaml	Окно с информацией о врачах и функцией удаления.
Deleteprosedurec.xaml	Окно с информацией о процедурах и функцией удаления и изменения.
Style.xaml	Стили

Папка Classes содержит в себе следующие классы:

1. BoolToVisibilityConverter.cs – класс-конвертер, преобразующий

логические значения в свойства видимости элементов интерфейса

2. Clients.cs – содержит информацию о конкретном пользователе;
3. Coupon.cs – содержит информацию о талонах;
4. Doctor.cs – содержит информацию о докторах;
5. Order.cs – содержит информацию о заказах;
6. Procedures.cs – содержит информацию о процедурах;
7. Review.cs – содержит информацию о заказах определенного

пользователя;

8. Schedule.cs – содержит информацию о расписании врача;
9. Spezialization.cs – содержит специальности.

Папка View содержит в себе следующие классы:

1. AppViewModel.cs – содержит реализацию команды закрытия окна через паттерн ICommand;
2. RelayCommand.cs – содержит саму реализацию паттерна ICommand;

3.3. Основные классы

Класс Login реализует методы, необходимые для авторизации пользователей и содержит следующие методы:

- переход на страницу регистрации;
- вход для авторизованного пользователя;
- сравнение логина при регистрации и авторизации;
- вход для гостя.

Класс Registration необходим для регистрации пользователей и содержит следующие методы:

- переход на страницу авторизации;
- регистрация;
- проверка на уникальность логина;
- валидация почты.

Класс MainHarmony предоставляет пользователю приветствие с обращением по имени, а также его аватар. Содержит следующие методы:

- вход в профиль пользователя (Profil.xaml);
- выход из учетной записи;
- переход на основные страницы программного средства (Главная, Врачи, Процедуры, Отзывы, о нас).

Класс Main представляет из себя главную страницу. Содержит следующие методы:

- переход на форму заказа талона (Createorder.xaml);
- переход на станицу с заказами(AllOrder.xaml);
- переход на страницу о нас (Aboutas.xaml).

Класс Profile содержит информацию о пользователе. Содержит следующие методы:

- обновить или добавить фотографию пользователя;

- изменить поля профиля: Имя, Фамилия, почта;
- обновить пароль.

Класс `allDoctor` представляет пользователю список врачей с кратким описанием. Содержит следующие методы:

- добавление врачей (`RegistrationofDoctor.xaml`);
- добавление талонов (`AddTalon.xaml`);
- удаление врачей (`DeleteDoctor.xaml`);
- фильтрация врачей по процедурам.

Класс `allprocedures` представляет пользователю список процедур с кратким описанием. Содержит следующие методы:

- добавление процедур (`RegistrationofProcedures.xaml`);
- удаление и редактирование процедур (`Deleteprosedurec.xaml`);
- фильтрация процедур по врачам.

Класс `Reviews` демонстрирует пользователю список отзывов. Содержит следующие методы:

- создание отзыва (`AddReviews.xaml`).

Класс `Adoutas` демонстрирует общую информацию об медцентре.

Класс `AddReviews` предоставляет возможность авторизованному пользователю оставлять отзывы.

Класс `AddTalon` предоставляет возможность администратору добавлять новые талоны определенному врачу на определенный промежуток дней.

Класс `RegistrationofDoctor` предоставляет возможность администратору регистрировать новых врачей, параллельно проводя проверку на уникальность данных и их валидацию.

Класс `RegistrationofProcedures` предоставляет возможность администратору регистрировать новые процедуры, параллельно проводя проверку на уникальность данных.

Класс `Createorder` предоставляет возможность авторизованному пользователю бронировать талоны, выбирая врача, процедуру, дату и время, после чего на указанную почту пользователю придет сообщение.

Класс `Yourorder` предоставляет возможность авторизованному пользователю просмотреть активные заказы и краткую информацию: Процедуру, лечащего врача, дату и время.

Класс `MainWindow` осуществляет переход на страницы авторизации и регистрации.

Класс `MessangeBox_Ок` реализует метод нажатия кнопки «ОК» и вывод определенного сообщения.

Класс `AllOrder` предоставляет возможность администратору просмотреть все активные заказы.

Класс `DeleteDoctor` предоставляет возможность администратору удалять докторов.

Класс `Deleteprosedurec` предоставляет возможность администратору удалять или же изменять процедуры.

3.4. Основные методы

Важным методом в работе с данным программным средством является регистрация. Листинг программного кода реализации процесса регистрации приведен в Приложении А.

Также важным является процесс авторизации пользователя, реализованный в классе Login. Листинг кода реализации процесса авторизации пользователя приведен в Приложении А.

3.5. Модель базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

- обеспечение хранения в БД всей необходимой информации;
- обеспечение возможности получения данных по всем необходимым запросам;
- сокращение избыточности и дублирования данных;
- обеспечение целостности базы данных.

Был выбран ADO.Net, так как он предоставляет самый прямой способ доступа к данным в .Net Framework.

Для реализации поставленной задачи была создана до начала написания самого приложения база данных MedicalCenter, которая включает в себя 9 таблиц: Order, Procedures, Clients, Reviews, Spezialization, Coupon, Doctor, schedule, Prescriptions. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. База данных MedicalCenter представлена в Приложении Б.

Таблица Client, содержащая информацию о пользователе, состоит из 7 столбцов, где поле «ID_Client» является первичным ключом. «ID_Client» – id клиента, поле Name_Client хранит имя пользователя, Surname_Client – фамилию, Email – почту, Password – пароль, Login – логин и Image – изображение.

Таблица «Review», содержащая информацию о отзывах, состоит из 4 столбцов. Она связана с таблицей «Clients» через поле «ID_Client». В данной таблице поле «ID_Review» является первичным ключом. «ID_Review» содержит id отзыва, поле «ID_Client» содержит id пользователя, совершившего заказ. Поле «Heading» – тема отзыва, «Review» – сам отзыв.

Таблица «Coupon» включает 5 столбцов. Она содержит информацию о талонах и связана с таблицей «Doctor» через поле «ID_Doctor», с таблицей «Order» - через поле «ID_Coupon». В данной таблице поле «ID_Coupon» является первичным ключом. Поле «ID_Coupon» содержит id талона, «Date» – дата талона, «Time» – время приема, «ID_Doctor»-id доктора, «Ordered»-поле, оповещающее, что талон забронирован или нет.

Таблица «Doctor», которая содержит информацию о врачах, состоит

из 10 столбцов. Она связана с таблицей «schedule» через поле «ID_Doctor», с таблицей «Spezialization» – через поле «Spezialization», с таблицей «Order» – через поле «ID_Doctor». В данной таблице поле «ID_Doctor» является первичным ключом. Поле «ID_Doctor» хранит id доктор, «Name_Doctor» содержит имя доктора, «Surname_Doctor» – фамилия доктора, «Middle_name_Doctor» – отчество доктора, «Spezialization» – специальность доктора, «Study» – ученная степень доктора, «Work_experience» – год начала карьеры, «Photo_Doctor» – фотография, «DoctorLogin» – логин доктора, «DoctorPassword» – пароль доктора.

Таблица «schedule», которая содержит информацию о расписании врача, состоит из 11 столбцов. Она связана с таблицей «Doctor» через поле «ID_doctor». Первичный ключ — «ID_doctor». Поля таблицы: «ID_doctor» – id доктора, «Mondfrom» – начало работы в понедельник, «Mondto» – окончание работы в понедельник, «Tuefrom» – начало работы во вторник, «Tueto» – окончание работы во вторник, «Wenfrom» – начало работы в среду, «Wento» – окончание работы в среду, «Thufrom» – начало работы в четверг, «Thuto» – окончание работы в четверг, «Frifrom» – начало работы в пятницу, «Frito» – окончание работы в пятницу.

Таблица «Order», которая содержит информацию о заявках, состоит из 8 столбцов. Она связана с таблицей «Clients» через поле «ID_Client», с таблицей «Coupon» – через поле «ID_Coupon», с таблицей «Doctor» – через поле «ID_Doctor», с таблицей «Procedures» – через поле «ID_Procedures». В данной таблице поле «Order_number» является первичным ключом. Поле «Order_number» содержит id заказа, «Date» – дата приема, «ID_Procedures» – id процедуры, «ID_Client» – id клиент, , «ID_Doctor» – id доктора, , «ID_Coupon» – id талона, , «Time» – время приема, «Status» – заявки.

Таблица «Procedures» включает 6 столбцов. Она содержит информацию о процедурах и связана с таблицей «Order» через поле «ID_Procedures», с таблицей «Spezialization» – через поле «Spezialization». В данной таблице поле «ID_Procedures» является первичным ключом. Поле «ID_Procedures» содержит id процедуры, «Name_Procedures» – название процедуры, «Decription» – описание, «Price» – цена процедуры, «Photo» – .фотография, «Spezialization» – специализация.

Таблица «Spezialization» содержит информацию о специальностях и включает 1 столбец. Она связана с таблицами «Doctor» и «Procedures» через поле «Spezialization». Первичный ключ — «Spezialization», который содержит название специальности.

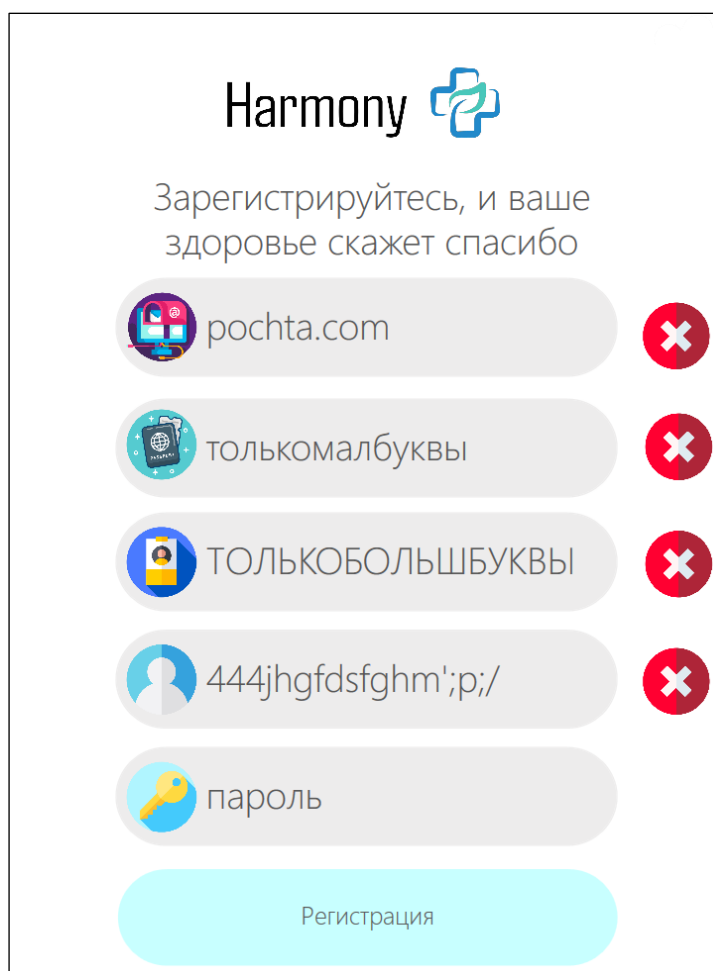
Таблица «Prescriptions», которая содержит информацию о рецептах, назначенных доктором. Она состоит из 5 столбцов и связана с таблицей «Doctor» через поле «DoctorID», с таблицей «Clients» - через поле «ClientID». В данной таблице поле «PrescriptionID» является первичным ключом. Поле «PrescriptionID» содержит id рецепта, «DoctorID» – id доктора, «ClientID» – id клиента, «Text» – текст рецепта, «Date» – дата назначения.

4. Тестирование

Данное программное средство тестировалось вручную, были выполнены все технические моменты, предусмотренные реализацией, а также были предприняты попытки нарушить работу приложения.

В курсовом проекте задействуется обработка исключений, таким образом, что пользователь будет уведомлен о неудачном выполнении операции. Присутствуют различные всплывающие окна, выводящие сообщение об ошибке, также присутствует просто текст-оповещение, а также различные изображения.

В момент регистрации возможна ситуация, когда пользователь вводит уже существующий логин или же просто некорректно вводит электронную почту, имя и фамилию. На рисунке 4.1 продемонстрирована обработка данных исключения.

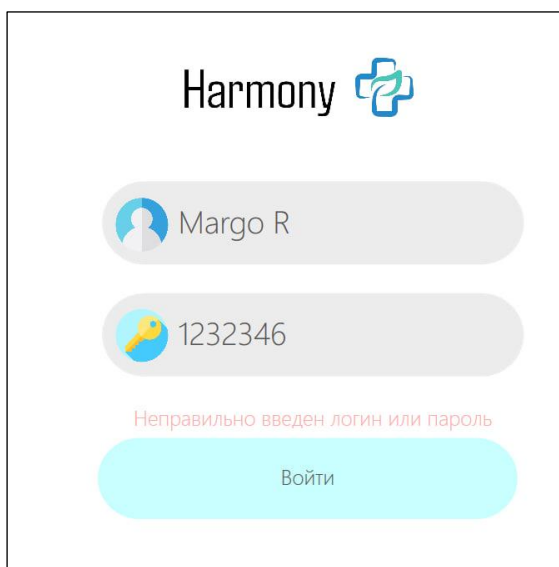


The image shows a registration form for an application named "Harmony". The form has a light blue header with the "Harmony" logo. Below the header, there is a message: "Зарегистрируйтесь, и ваше здоровье скажет спасибо". The form contains five input fields, each with an icon on the left and a red circle with a white 'X' on the right, indicating a validation error. The fields are: 1. Email: "pochta.com" with a red 'X'. 2. Username: "толькомалбуквы" with a red 'X'. 3. Username: "ТОЛЬКОБОЛЬШБУКВЫ" with a red 'X'. 4. Username: "444jhgfdsfghm';p;/" with a red 'X'. 5. Password: "пароль" with a red 'X'. At the bottom of the form is a large blue button labeled "Регистрация".

Рисунок 4.1 – Неверно введен почта, имя, фамилия и логин

При регистрации нового пользователя его данные заносятся в таблицу Clients в базе данных. При попытке входа в учетную запись, введенные логин и пароль проверяются с занесенными в базу данных. В случае неправильного ввода логина или пароля выводится сообщение,

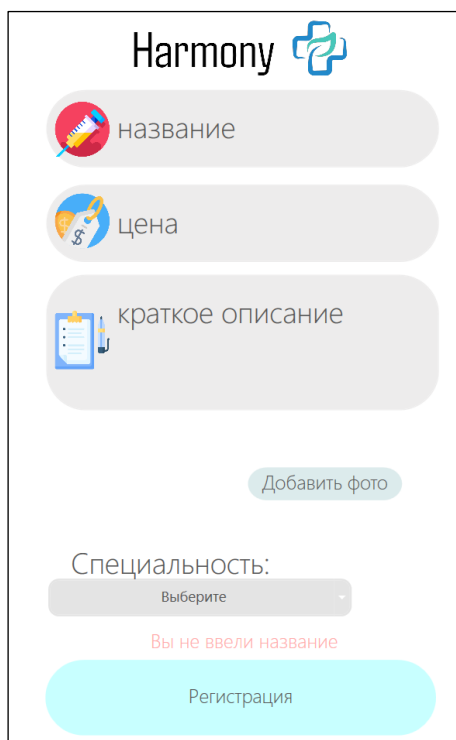
представленное на рисунке 4.2.



The screenshot shows the Harmony login interface. At the top is the 'Harmony' logo with a blue and green cross icon. Below it are two input fields: the first contains a user icon and the text 'Margo R'; the second contains a key icon and the text '1232346'. A red error message 'Неправильно введен логин или пароль' is displayed below the password field. At the bottom is a large cyan button labeled 'Войти'.

Рисунок 4.2 – Неверный ввод логина или пароля при входе

Возможна ситуация, когда администратор при создании процедуры заполнил не все поля, тогда программное средство напомнит ему об этом и не даст создать процедуру, пока администратор не исправит все свои ошибки и не заполнит все поля. Демонстрация обработки данной ошибки представлена на рисунке 4.3.



The screenshot shows the Harmony procedure creation interface. At the top is the 'Harmony' logo. Below it are three input fields: the first has a syringe icon and the text 'название'; the second has a dollar sign icon and the text 'цена'; the third has a clipboard icon and the text 'краткое описание'. Below these fields is a cyan button labeled 'Добавить фото'. Further down is a dropdown menu labeled 'Специальность:' with the text 'Выберите' and a downward arrow. A red error message 'Вы не ввели название' is displayed below the dropdown. At the bottom is a large cyan button labeled 'Регистрация'.

Рисунок 4.3 – Заполнены не все поля

Также при создании процедуры администратор может указать

название процедуры, которое уже существует. Этот случай также предусмотрен и продемонстрирован на рисунке 4.4.

Рисунок 4.4 – Заполнены не все поля

Аналогичные проверки имеются и при создании администратором врача.

На рисунке 4.5 изображена ошибка, когда администратор при создании врача заполнил не все поля.

Рисунок 4.5 – Заполнены не все поля

На рисунке 4.6 изображена ошибка, когда администратор указал имя врача, которое уже существует.

Рисунок 4.6 – Заполнены не все поля

При заказе талона пользователь может не ввести определенные данные и попытаться завершить заказ, но и данная ошибка тоже обрабатывается, и до того момента, пока все поля не будут заполнены, будут всплывать ошибки, что помогут корректно завершить оформление талона. Данная обработка ошибки изображена на рисунке 4.7.

Рисунок 4.7 – Заполнены не все поля

Так же в данном программном средстве существует функция по изменению пароля, когда пользователь ввел неправильный нынешний пароль появляется оповещение, изображенное на рисунке 4.8.

Рисунок 4.8 – Введен неверный нынешний пароль

Пользователь здесь же может преднамеренно или же по ошибке указать два одинаковых пароля. Этот случай также предусмотрен и продемонстрирован на рисунке 4.9

Рисунок 4.9 – Ввод одного и того же пароля

Были продемонстрированы все возможные ситуации неверной эксплуатации программного средства пользователем, проведена проверка работоспособности и проанализированы полученные результаты.

4.1. Руководство по использованию

Для того чтобы в полной мере пользоваться приложением, необходимо для начала зарегистрироваться в системе. Окно регистрации показано на рисунке 4.10.

Рисунок 4.10 – Окно регистрации

Если у пользователя уже есть аккаунт, то нужно войти в систему. Окно входа показано на рисунке 4.11.

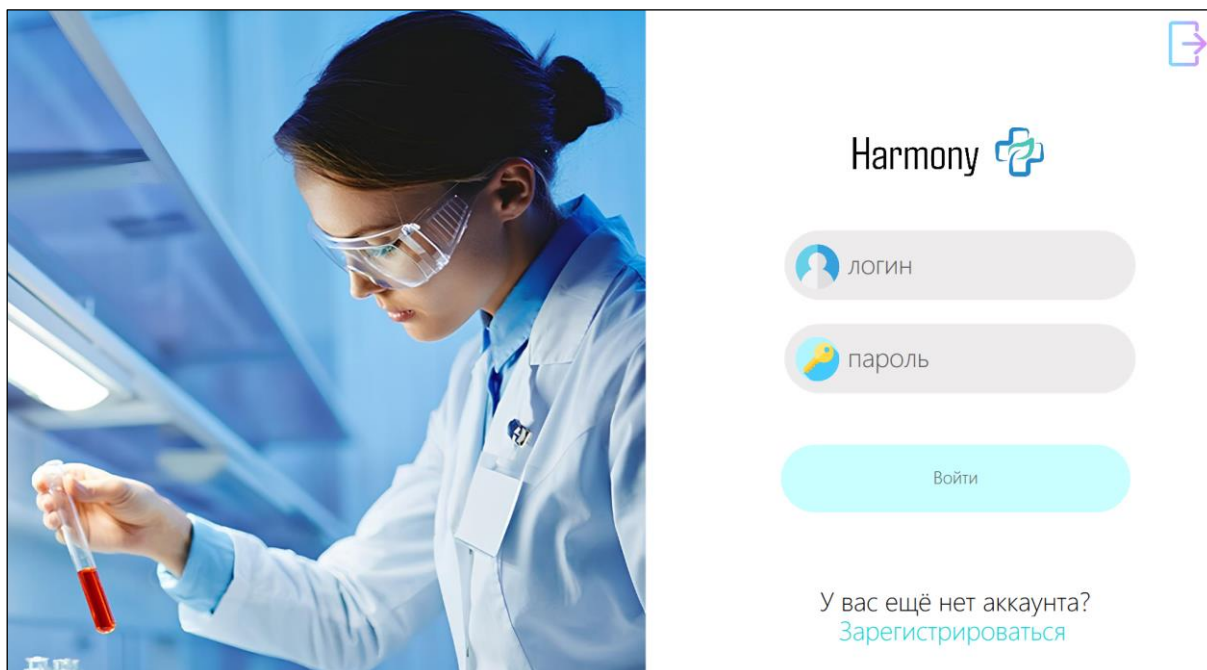


Рисунок 4.11 – Окно авторизации

4.1.1. Руководство по использованию администратором

После входа в систему администратору откроется главное окно. В левой части окна находятся активные кнопки для перехода по различным страницам. Главное окно представлено на рисунке 4.12

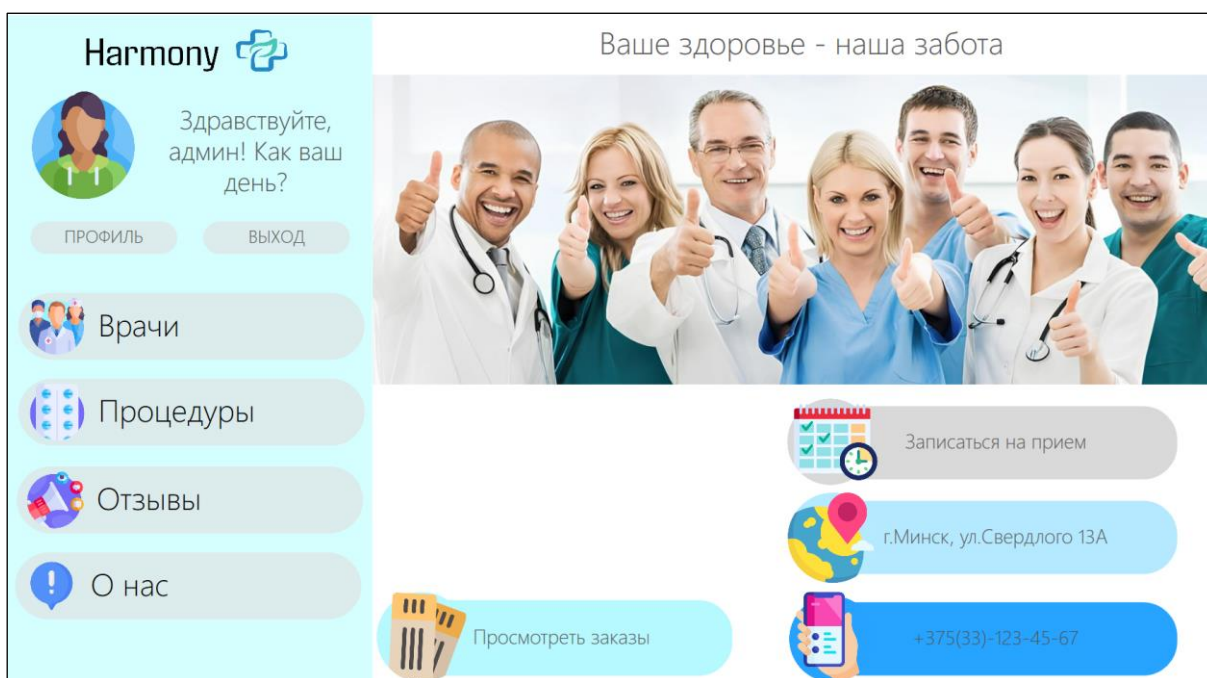


Рисунок 4.12 – Главное окно

Нажав на кнопку, «Врачи» мы переходим к общему перечню врачей, которых мы можем отсортировать по выполняемым процедурам. Так же для администратора имеются две дополнительные кнопки «добавить талон» и плюс, с помощью которого можно добавить нового врача в базу данных. Представлено на рисунке 4.13.

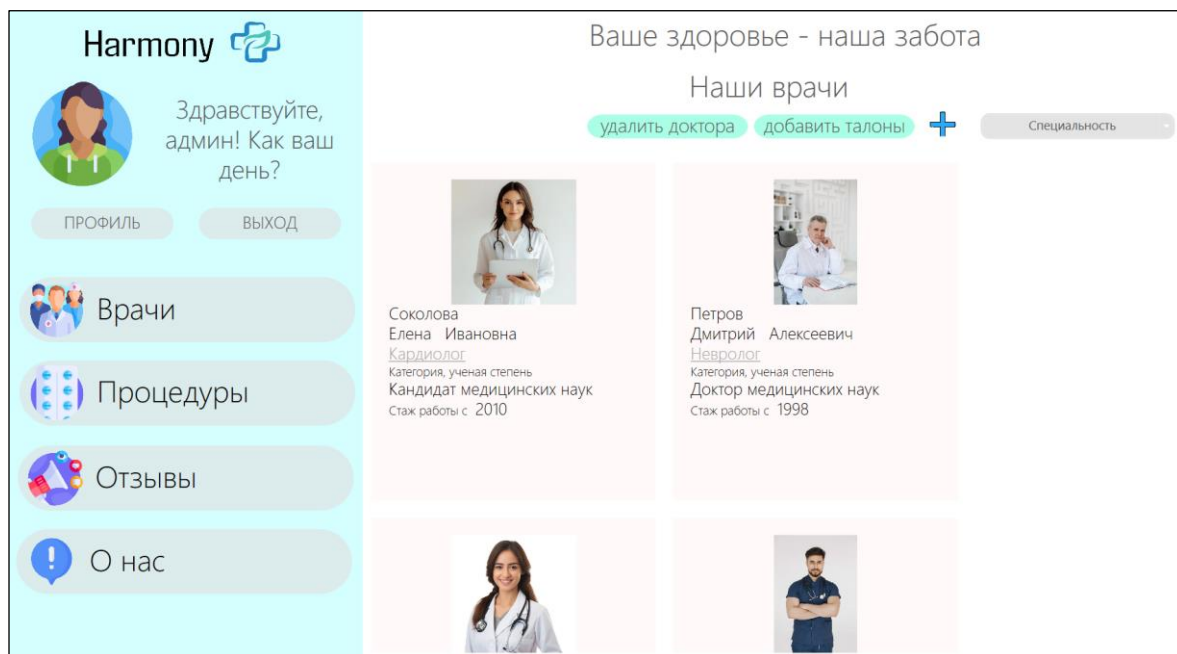


Рисунок 4.13 – Окно с перечнем врачей

Нажав на кнопку, «Процедуры» мы переходим к общему перечню процедур, которых мы можем отсортировать по врачам. Так же для администратора имеется дополнительная кнопка «удалить/изменить». Представлено на рисунке 4.13.

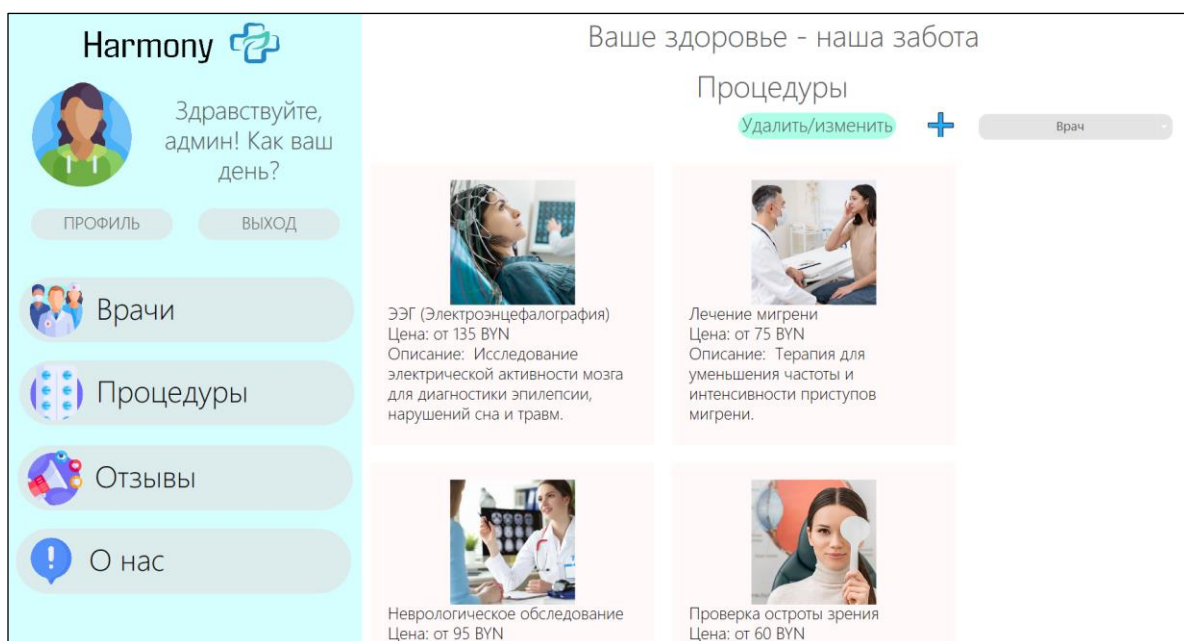


Рисунок 4.13 – Окно с перечнем процедур

4.1.2. Руководство по использованию пользователем

После входа в систему пользователю откроется тоже самое главное окно, откуда мы можем сразу же заказать талон, нажав по кнопке «Записаться на прием», где мы можем выбрать процедуру или врача, по которому будет происходить фильтрация, после уже пользователю выдаются не забронированные дни и время. По нажатию кнопки «Заказать», пользователю всплывает сообщение о том, что заказ принят, а на почту пришло сообщение о подтверждении брони. Представлено на рисунке 4.14-4.15.

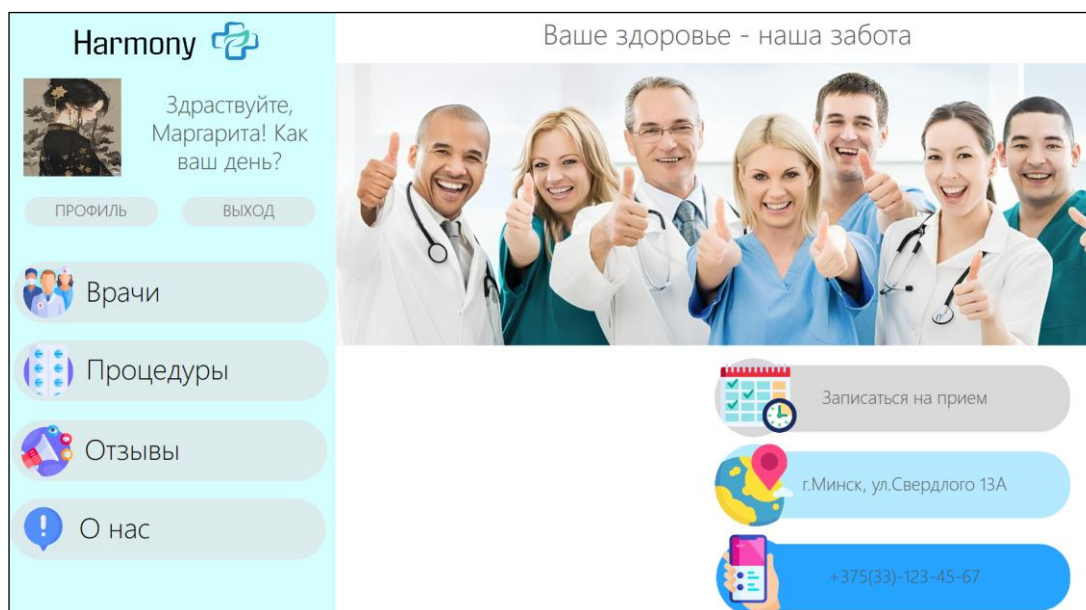


Рисунок 4.14 – Главное окно

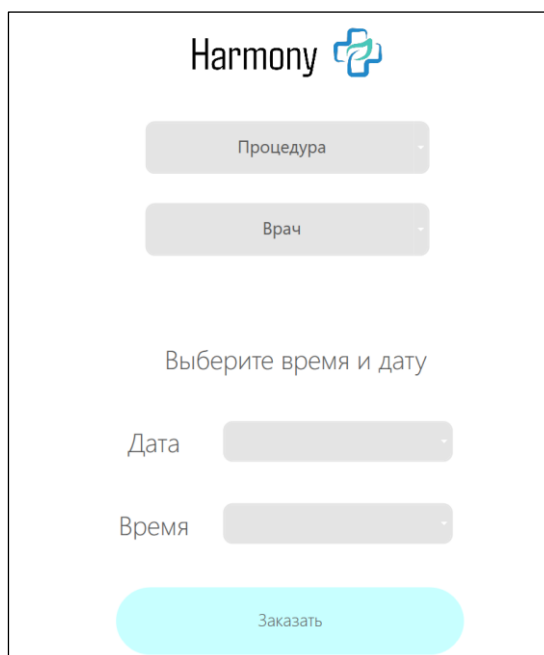


Рисунок 4.15 – Окно регистрации заказа

Нажав на кнопку, «Врачи» мы переходим к общему перечню врачей, которых мы можем отсортировать по выполняемым процедурам. Как видно, кнопки для администратора скрылись. Представлено на рисунке 4.16.

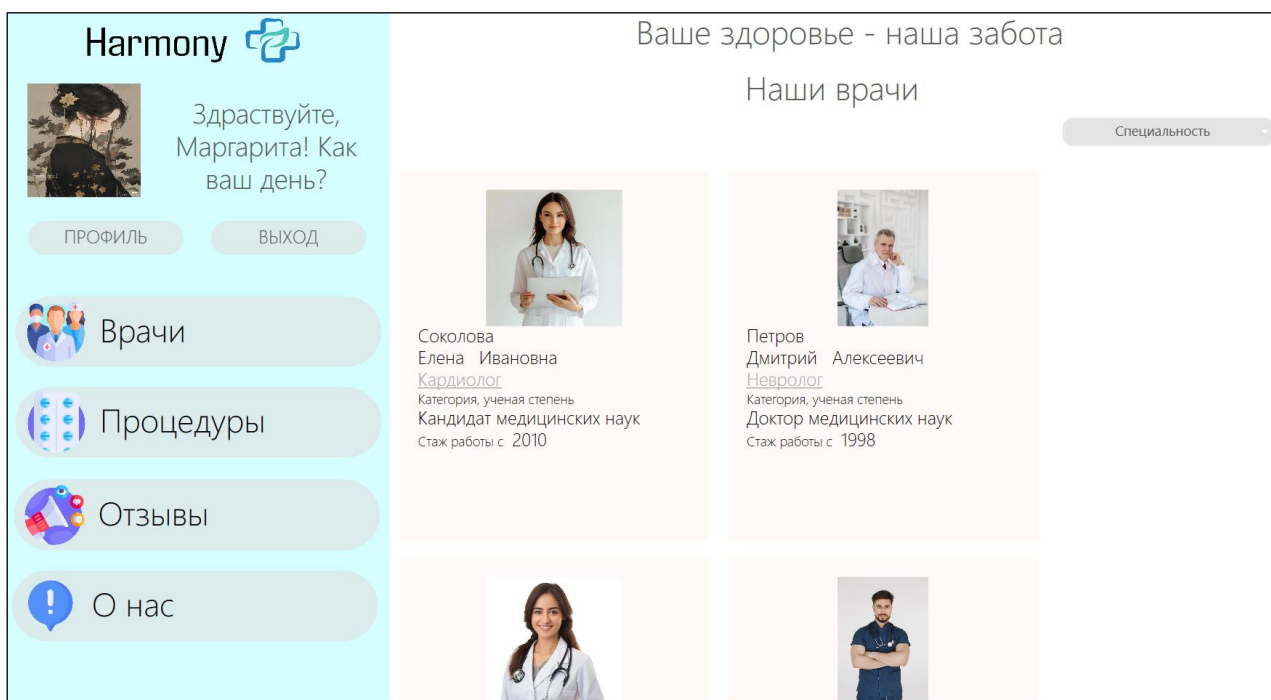


Рисунок 4.16 – Окно с перечнем врачей

Нажав на кнопку, «Процедуры» мы переходим к общему перечню процедур, которых мы можем отсортировать по врачам. Как видно, кнопки для администратора скрылись. Представлено на рисунке 4.17.

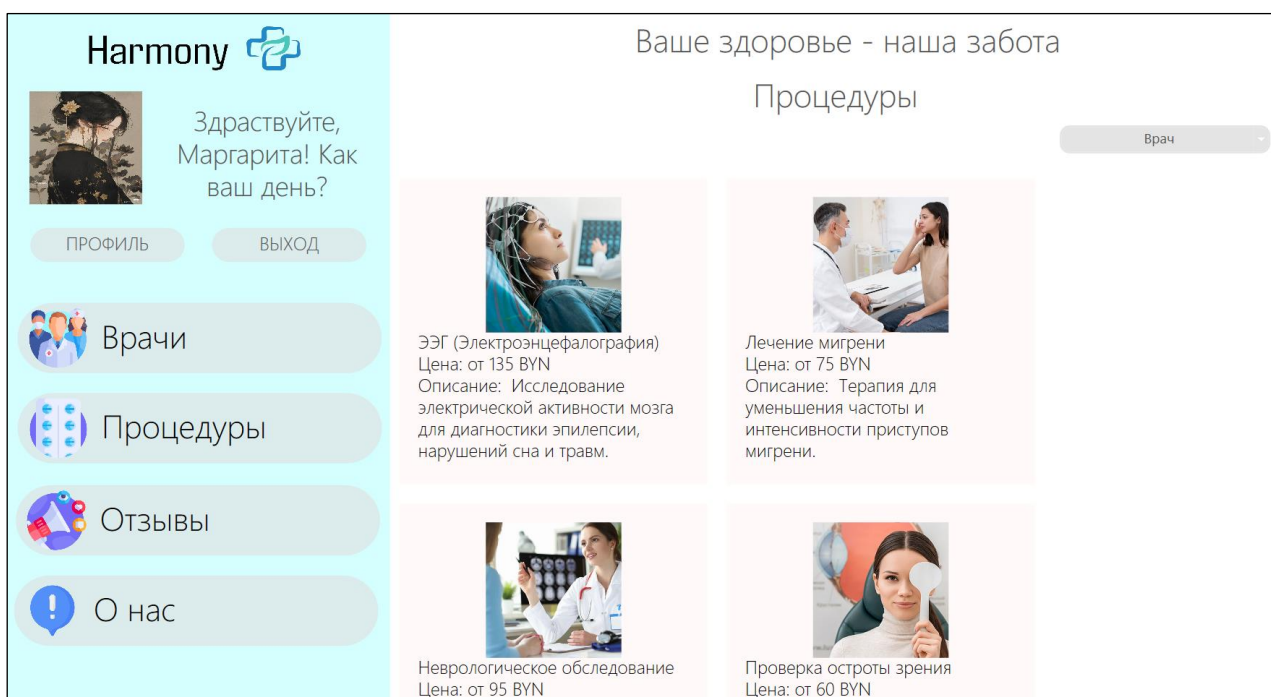


Рисунок 4.17 – Окно с перечнем процедур

4.1.3. Руководство по использованию врачом

После входа в систему врачу откроется главное окно, где он сможет управлять своими записями и взаимодействовать с пациентами. Вот основные функции, доступные врачу: назначение рецептов, подтверждение оказания услуги. Представлено на рисунке 4.18.

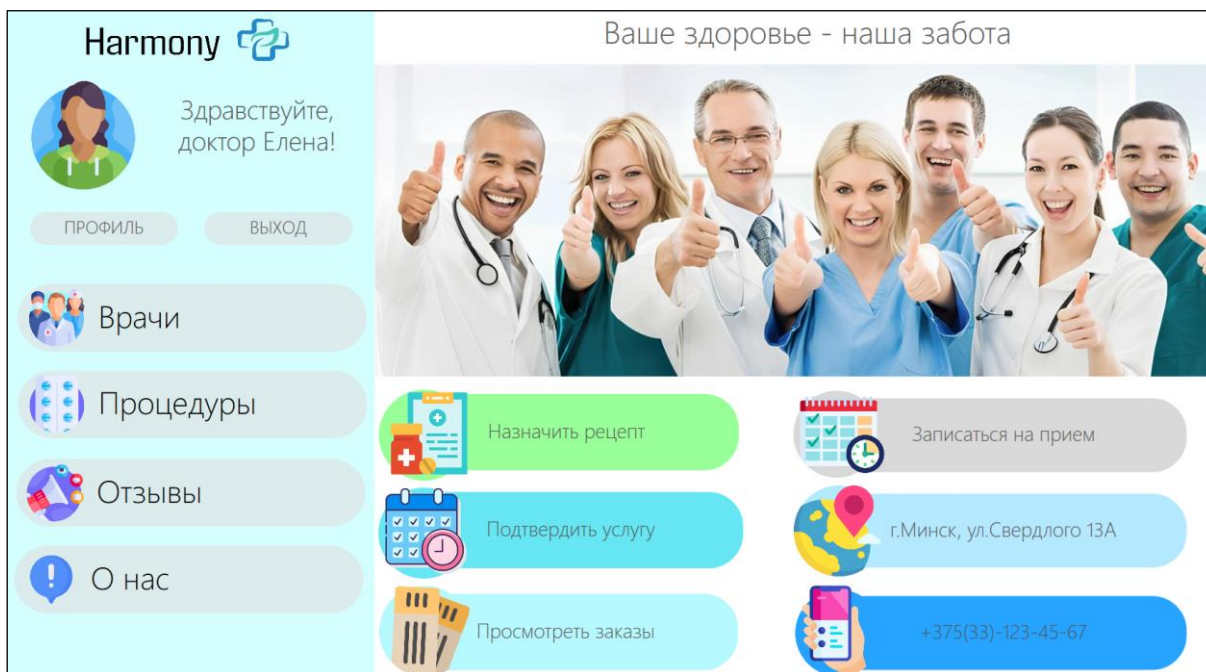


Рисунок 4.18 – Главное окно

Нажав на кнопку, «Назначить рецепт» откроется окно, представленное на рисунке 4.19.

Рисунок 4.19 – Окно назначение рецепта

Нажав на кнопку, «Подтвердить услугу» откроется окно, представленное на рисунке 4.19.

Ваше здоровье - наша забота

Активные заказы всех клиентов

Номер заказа	ID Врача	Врач	Цена	Купон	Дата	Время	ID Клиент	Клиент	Процедура	Подтверждение	Отмена
21	1	Елена Иванов	150	2040	19.05.2025	08:30:00	2	Мargarита Р	Холтеровское мс	Подтвердить оказание	Отмена

Рисунок 4.19 – Окно подтверждение услуги

Приложение обеспечивает эффективное взаимодействие между администраторами, врачами и пациентами, упрощая процесс записи на прием и управление медицинскими процедурами.

Заключение

В результате выполнения данного курсового проекта было создано программное средство под названием «MedicalCenter». В процессе разработки были учтены все пункты из списка предполагаемого основного функционала приложения.

В программном обеспечении были реализованы следующие функции: регистрация пользователей в системе;

- вход пользователей в систему;
- вход гостей в систему;
- просмотр информации о врачах, процедурах, отзывах, медицинском центре и забронированных талонах;
- добавление информации о врачах и процедурах;
- удаление информации о врачах и процедурах;
- редактирование процедур;
- сортировка врачей по процедурам и сортировка процедур по врачам;
- бронирование талонов;
- добавление отзывов;
- регистрация новых талонов;
- назначение рецептов;
- обработка заявок;
- изменение информации в личном кабинете.

Согласно полученным результатам работы программы, можно сделать вывод, что разработанное приложение функционирует корректно, а требования технического задания выполнены в полном объеме.

Список использованных источников

1. Сайт медицинского центра «Эксана» [Электронный ресурс] / Режим доступа: <https://eksana.by/> . Дата доступа: 10.03.2025
2. Сайт медицинского центра «Клиника Каскад» [Электронный ресурс] / Режим доступа: <https://kaskadclinic.by/> . Дата доступа: 10.03.2025
3. Сайт медицинского центра «ЛЮДЭ» [Электронный ресурс] / Режим доступа: <https://www.lode.by/> . Дата доступа: 10.03.2025
4. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> . Дата доступа: 12.04.2025
5. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 13.04.2025
6. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> . Дата доступа: 15.04.2025

Приложение А

Листинг 1: Класс Login

```
public partial class Login : Page
{
    string connectionString;
    public Login()
    {
        InitializeComponent();
        connectionString =
ConfigurationManager.ConnectionStrings["MedicalCenter.Properties.Settings.Corser_projectConnectionString"].ConnectionString;
    }

    private void Password_GotFocus(object sender,
RoutedEventArgs e)
    {
        if (Password.Text == "пароль")
        {
            Password.Text = "";
        }
    }

    private void Password_LostFocus(object sender,
RoutedEventArgs e)
    {
        if (string.IsNullOrEmpty(Password.Text))
            Password.Text = "пароль";
    }

    private void Login_GotFocus(object sender, RoutedEventArgs
e)
    {
        if (Logine.Text == "ЛОГИН")
        {
            Logine.Text = "";
        }
    }

    private void Login_LostFocus(object sender,
RoutedEventArgs e)
    {
        if (string.IsNullOrEmpty(Logine.Text))
            Logine.Text = "ЛОГИН";
    }

    private void TextBlock_MouseDown(object sender,
MouseButtonEventArgs e)
    {
        Manager.MainFrame.Navigate(new Registration());
    }
}
```

```

    }

    private void Button_Click(object sender, RoutedEventArgs
e)
    {
        // 1. Проверка администратора
        bool pointadmin = false;
        ForAdmin.Admin admin = new ForAdmin.Admin();
        if (Logine.Text == admin.login && Password.Text ==
admin.password)
        {
            MainHarmony window1 = new MainHarmony(admin);
            window1.Show();
            foreach (Window window in
Application.Current.Windows)
            {
                if (window is MainWindow)
                {
                    window.Close();
                    pointadmin = true;
                    break;
                }
            }
        }
        // 2. Проверка врача
        string doctorSql = "SELECT * FROM Doctor WHERE
DoctorLogin = @Login AND DoctorPassword = @Password";
        using (SqlConnection doctorConnection = new
SqlConnection(connectionString))
        {
            doctorConnection.Open();
            SqlCommand doctorCommand = new
SqlCommand(doctorSql, doctorConnection);
            doctorCommand.Parameters.AddWithValue("@Login",
Logine.Text);
            doctorCommand.Parameters.AddWithValue("@Password",
Password.Text);

            using (SqlDataReader doctorReader =
doctorCommand.ExecuteReader())
            {
                if (doctorReader.HasRows)
                {
                    Classes.Doctor doctor = new
Classes.Doctor();
                    while (doctorReader.Read())
                    {
                        doctor.ID_Doctor =
doctorReader["ID_Doctor"];
                        doctor.Name_Doctor =
doctorReader["Name_Doctor"].ToString();

```

```

        doctor.Surname_Doctor =
doctorReader["Surname_Doctor"].ToString();
        doctor.DoctorLogin =
doctorReader["DoctorLogin"].ToString();
        doctor.DoctorPassword =
doctorReader["DoctorPassword"].ToString();
    }
    MainHarmony window1 = new
MainHarmony(doctor);
    window1.Show();
    Application.Current.Windows[0].Close();
    return;
    }
    }
    }
    // 3. Проверка клиента
    string sql = "SELECT * FROM Clients WHERE Login =
@Login AND Password = @Password"; // Используйте
параметризованный запрос
    try
    {
        using (SqlConnection sqlConnection = new
SqlConnection(connectionString))
        {
            sqlConnection.Open();
            if (Logine.Text != "логин" && Password.Text !=
"пароль")
            {
                SqlCommand sqlCommand = new
SqlCommand(sql, sqlConnection);

sqlCommand.Parameters.AddWithValue("@Login", Logine.Text);

sqlCommand.Parameters.AddWithValue("@Password",
Password.Text);

                using (SqlDataReader reader =
sqlCommand.ExecuteReader())
                {
                    Classes.Clients clients = new
Classes.Clients();

                    if (reader.HasRows)
                    {
                        bool flagClients = false;
                        while (reader.Read())
                        {
                            // Проверка пароля и логина
                            if (Logine.Text ==
reader["Login"].ToString() && Password.Text ==
reader["Password"].ToString())
                            {
                                flagClients = true;

```

```

reader["ID_Client"];
reader["Name_Client"].ToString();
reader["Surname_Client"].ToString();
reader["Email"].ToString();
reader["Password"].ToString();
reader["Login"].ToString();

DBNull.Value)

(byte[])reader["Image"];

GetDefaultImageBytes();

clients.ID_Clients =
clients.Name_Client =
clients.Surname_Client =
clients.Email =
clients.Password =
clients.Login =

// Обработка изображения
if (reader["Image"] !=
{
    clients.Image =
}
else
{
    clients.Image =
}

break;
}
}
reader.Close();

if (flagClients)
{
    MainHarmony window1 = new
    window1.Show();
    foreach (Window window in
Application.Current.Windows)
    {
        if (window is MainWindow)
        {
            window.Close();
            break;
        }
    }
}
else
{
    if (!pointadmin)
        Warring.Text =
"Неправильно введен логин или пароль";
}

```



```

        }
    }
    else
    {
        Warring.Text = "Данные до конца не
введены";
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка: {ex.Message}");
}
}

private byte[] GetDefaultImageBytes()
{
    try
    {
        string defaultImagePath =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"image", "user.png");
        if (File.Exists(defaultImagePath))
        {
            return File.ReadAllBytes(defaultImagePath);
        }
        return new byte[0]; // Пустой массив, если файла
нет
    }
    catch
    {
        return new byte[0]; // Пустой массив в случае
ошибки
    }
}
}

```

Листинг 2: Класс Registration

```

public partial class Registration : Page
{
    private readonly string _basePath =
AppDomain.CurrentDomain.BaseDirectory;
    public bool emailbool = true;
    public bool loginregistr = false;
    string connectionString;
    bool names = true;
    bool surnames = true;
    bool password = true;
    string allowedchar = "";
    public Registration()
    {

```

```

        InitializeComponent();
        connectionString =
ConfigurationManager.ConnectionStrings["MedicalCenter.Properties.Settings.Corser_projectConnectionString"].ConnectionString;
    }

    private void email_GotFocus(object sender, RoutedEventArgs
e)
    {
        if (email.Text == "электронная почта")
        {
            email.Text = "";
        }
    }

    private void email_LostFocus(object sender,
RoutedEventArgs e)
    {
        if (string.IsNullOrEmpty(email.Text))
            email.Text = "электронная почта";
        string emails = email.Text;
        Regex regex = new Regex(@"^([\w\.\-]+)@([\w\.-
]+)((\.(\w){2,3})+)$");
        Match match = regex.Match(emails);
        if ((match.Success))
        {
            string imagePath =
System.IO.Path.Combine(_basePath, "image", "yes.png");
            Yesornoemail.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
            emailbool = true;
        }
        if ((match.Success == false))
        {
            string imagePath =
System.IO.Path.Combine(_basePath, "image", "no.png");
            Yesornoemail.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
            emailbool = false;
        }
    }

    private void name_LostFocus(object sender, RoutedEventArgs
e)
    {
        if (string.IsNullOrEmpty(name.Text))
            name.Text = "имя";

        Regex regex = new Regex(@"^[А-ЯЁ][а-яё]+$");
        if (!regex.IsMatch(name.Text))
        {

```

```

        string imagePath =
System.IO.Path.Combine(_basePath, "image", "no.png");
        Yesornoname.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
        names = false;
    }
    else if (regex.IsMatch(name.Text))
    {
        string imagePath =
System.IO.Path.Combine(_basePath, "image", "yes.png");
        Yesornoname.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
        names = true;
    }
}

private void name_GotFocus(object sender, RoutedEventArgs
e)
{
    if (name.Text == "ИМЯ")
    {
        name.Text = "";
    }
}

private void Logine_GotFocus(object sender,
RoutedEventArgs e)
{
    if (Logine.Text == "ЛОГИН")
    {
        Logine.Text = "";
    }
}

private void Logine_LostFocus(object sender,
RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(Logine.Text))
    {
        Logine.Text = "ЛОГИН";
        return;
    }

    if (Logine.Text.Contains(allowedchar))
    {
        string imagePath =
System.IO.Path.Combine(_basePath, "image", "no.png");
        Yesornologine.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
        loginregistr = false;
        return;
    }
}

```

```

        string sql = "SELECT Login FROM Clients WHERE Login =
@Login";
        try
        {
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand command = new
SqlCommand(sql, connection))
                {
                    command.Parameters.AddWithValue("@Login",
Logine.Text);
                    using (SqlDataReader reader =
command.ExecuteReader())
                    {
                        if (reader.HasRows || Logine.Text ==
"admin")
                        {
                            string imagePath =
System.IO.Path.Combine(_basePath, "image", "no.png");
                            Yesornologine.Source = new
BitmapImage(new Uri(imagePath, UriKind.Absolute));
                            loginregistr = false;
                        }
                        else
                        {
                            string imagePath =
System.IO.Path.Combine(_basePath, "image", "yes.png");
                            Yesornologine.Source = new
BitmapImage(new Uri(imagePath, UriKind.Absolute));
                            loginregistr = true;
                        }
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка проверки логина:
{ex.Message}");
        }
    }

    private void Password_GotFocus(object sender,
RoutedEventArgs e)
    {
        if (Password.Text == "пароль")
        {
            Password.Text = "";
        }
    }

```

```

    }

    private void Password_LostFocus(object sender,
RoutedEventArgs e)
    {
        if (string.IsNullOrEmpty(Password.Text))
            Password.Text = "пароль";

        if ((Password.Text.ToString().Contains(allowedchar)))
        {
            string imagePath =
System.IO.Path.Combine(_basePath, "image", "no.png");
            Yesornopassword.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
            password = false;
            return;
        }
    }

    private void Image_MouseDown(object sender,
MouseButtonEventArgs e)
    {
        Manager.MainFrame.Navigate(new Login());
    }
    private void Button_Click(object sender, RoutedEventArgs
e)
    {
        // Проверка всех условий
        if (!emailbool)
        {
            MessageBox.Show("Некорректный email!");
            return;
        }
        if (!loginregistr)
        {
            MessageBox.Show("Логин уже занят или содержит
запрещённые символы!");
            return;
        }
        if (!password || Password.Text.Length < 6)
        {
            MessageBox.Show("Пароль должен быть не менее 6
символов и не содержать апостроф!");
            return;
        }
        if (!names || !surnames)
        {
            MessageBox.Show("Имя и фамилия должны быть на
кириллице!");
            return;
        }
    }

```

```

        byte[] imageBytes;
        try
        {
            string imagePath =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"image", "woman.png");
            imageBytes = File.ReadAllBytes(imagePath);
        }
        catch
        {
            imageBytes = null;
        }
        string sql = @"
INSERT INTO Clients
    (Name_Client, Surname_Client, Email, Login, Password,
Image)
VALUES
    (@Name, @Surname, @Email, @Login, @Password, @Image)";

        try
        {
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                connection.Open();
                using (SqlCommand command = new
SqlCommand(sql, connection))
                {
                    command.Parameters.Add("@Name",
SqlDbType.NVarChar).Value = name.Text;
                    command.Parameters.Add("@Surname",
SqlDbType.NVarChar).Value = Surname.Text;
                    command.Parameters.Add("@Email",
SqlDbType.NVarChar).Value = email.Text;
                    command.Parameters.Add("@Login",
SqlDbType.NVarChar).Value = Logine.Text;
                    command.Parameters.Add("@Password",
SqlDbType.NVarChar).Value = Password.Text;
                    command.Parameters.Add("@Image",
SqlDbType.VarBinary).Value = imageBytes != null ? imageBytes :
(object)DBNull.Value;
                    int rowsAffected =
command.ExecuteNonQuery();
                    if (rowsAffected > 0)
                    {
                        MessageBox_Ok message = new
MessageBox_Ok("Регистрация успешна!");
                        message.Show();
                        Manager.MainFrame.Navigate(new
Login());
                    }
                }
            }
        }
    }
}

```

```

    }
    catch (SqlException ex)
    {
        MessageBox.Show($"Ошибка базы данных:
{ex.Message}");
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}");
    }
}

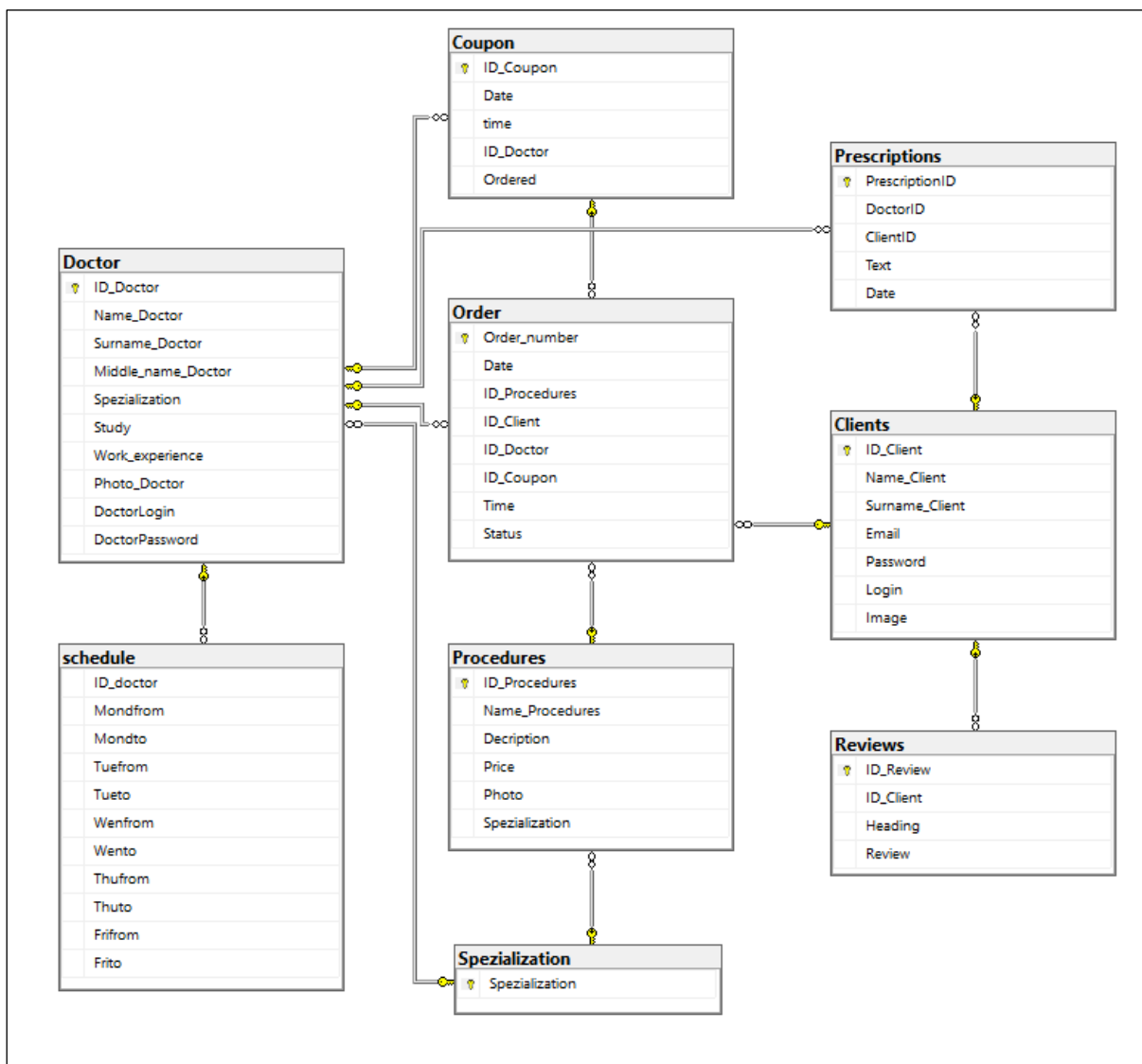
private void Surrename_GotFocus(object sender,
RoutedEventArgs e)
{
    if (Surrename.Text == "фамилия")
    {
        Surrename.Text = "";
    }
}

private void Surrename_LostFocus(object sender,
RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(Surrename.Text))
        Surrename.Text = "фамилия";

    Regex regex = new Regex(@"^[А-ЯЁ][а-яё]+$");
    if (!regex.IsMatch(Surrename.Text))
    {
        string imagePath =
System.IO.Path.Combine(_basePath, "image", "no.png");
        Yesornolastname.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
        surnames = false;
    }
    else if (regex.IsMatch(Surrename.Text))
    {
        string imagePath =
System.IO.Path.Combine(_basePath, "image", "yes.png");
        Yesornolastname.Source = new BitmapImage(new
Uri(imagePath, UriKind.Absolute));
        surnames = true;
    }
}
}

```

Приложение Б



Приложение В

