**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Дуларидзе Тынатын Тамазовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Дуларидзе Тынатын Тамазовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения. Проектирование UML-диаграмм;
* Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;
* Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 4**

Задание выдал «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Дуларидзе Т.Т.(подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Дуларидзе Тынатын Тамазовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 23.09.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 24.09.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 25.09.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 26.09.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 27.09.2024 | Проектирование интерфейса пользователя. |  |
| 28.09.2024 | Разработка схемы алгоритма программного продукта. |  |
| 30.09.2024 | Разработка модели данных. |  |
| 01.10.2024 | Разработка программного продукта. Создание функциональности ПО. |  |
| 02.10.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 03.10.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 04.10.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. |  |
| 05.10.2024 | Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. Подготовка отчета. |  |

СОДЕРЖАНИЕ

Введение 6

Задание №1 Разработка технического задания 7

Задание №2 Разработка алгоритмов и диаграмм 10

Задание №3 Разработка макетов программы 18

Задание №4 Разработка программы 22

Задание №5 Тестирование 25

Задание №1 Git hub. Оценка проекта 32

Заключение 33

Список используемых источников 34

Приложение А 35

Приложение Б 38

ВВЕДЕНИЕ

В рамках прохождения практики была поставлена задача провести комплексный анализ и разработку программного продукта, включающего в себя несколько ключевых этапов. Основные задачи, выполненные в ходе практики, включали анализ предметной области, методов решения и средств, исследовательские работы, разработку технического задания, проектирование UML-диаграмм, моделирование структуры программного обеспечения, проектирование интерфейса пользователя, разработку алгоритмов и моделей данных, а также создание и тестирование программного продукта.

Целью практики было не только освоить теоретические аспекты разработки программного обеспечения, но и применить их на практике, что позволило бы углубить понимание процессов разработки и улучшить навыки в области программирования и проектирования. В ходе выполнения заданий были использованы современные методы и инструменты, что позволило достичь высокого уровня качества и надежности разработанного программного продукта.

Задание №1 Разработка технического задания

Организация АНО «Здоровье Плюс» предоставляет разного рода платные медицинские услуги. Мы заботимся о том, чтобы данные наших пациентов не были утеряны и украдены сторонними организациями, оказанные услуги прописаны в отдельных документах. Каждые год в последнюю неделю декабря мы устраиваем спонсорские мероприятия, где рассказываем о политике организации, наших достижениях, и планах развития в научно-медицинской сфере.

Для большей производительности нам необходима внутренняя система учета данных наших пациентов и оказанных им платных услуг.

Основной целью данной информационной системы является автоматизация процесса учета и обработки данных пациентов поликлиники. Система позволяет врачам эффективно заполнять все данные пациента, оказанные ему услуги.

Разработка информационной системы для частной клиники — это процесс создания системы, которая позволяет клинике эффективно вводить данные своих пациентов и отслеживать предоставленные им услуги.

Задача разработки такой системы состоит в создании удобного и эффективного инструмента для учета, записи пациентов на прием, отслеживания и управления данными пациентов, что помогает минимизировать время передачи данных о пациентах и увеличить скорость их обработки.

Основные функции и возможности системы включают:

1. Авторизацию пользователей: системный администратор предоставляет логин и пароли всем пользователям системы: врачам, регистраторам, самим админам.

2. Добавление новых пациентов в систему: это информация, которую указывает регистратор в системе: ФИО пациента, дата рождения, пол, рост, вес, паспортные данные, почта.

3. Просмотр данных своих пациентов: врач имеет доступ к данным пациентов.

4. Просмотр данных врачей: регистратор имеет возможность просматривать данные врачей: ФИО врача, профиль, кабинет.

5. Добавление записи к врачу: регистратор может записывать пациентов на прием.

6. Редактирование данных врачей: администратор системы может менять, добавлять, удалять данные о сотрудниках и услугах врачей.

В ходе анализа предметной обрасти было составлено техническое задание:

1. Общие сведения

1.1. Наименование проекта: Разработка информационной системы для учета и обработки данных в частной поликлинике.

1.2. Заказчик: АНО «Здоровье Плюс».

1.3. Исполнитель: «SoftUP» Development Team.

2. Функциональные требования

2.1. Возможность авторизации пользователей;

2.2. Возможность заполнения данных новых пациентов, и редактирования данных зарегистрированных пациентов для регистраторов:

- ФИО пациента;

- Дата рождения;

- Пол;

- Рост;

- Вес;

- Паспортные данные;

- Почта пациента.

2.3. Возможность оформления записи на прием к врачу для регистратора с указанием следующих данных:

- ФИО пациента;

- Наименованием медицинской услуги;

- Время записи.

2.4. Возможность просмотра данных для регистраторов о услугах;

2.5. Возможность просмотра данных пациентов для врачей;

2.6. Возможность добавления результатов после приема для врачей;

2.7. Возможность просмотра данных о записях на прием для врача;

2.8. Возможность редактирования сотрудников, ролей и услуг для администратора;

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС Windows 10 – 11.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Группировка информации и функций по важности и смыслу;

3.4. Производительность:

- Время отклика на запросы пользователя менее 5 секунд.

4. Требования к реализации

4.1. Язык программирования: C#

4.2. СУБД: Microsoft SQL Server

5. Требования к документации

5.1. Техническое задание на разработку информационной системы.

6. Руководство по стилю

6.1. Шрифт: Arial

6.2. Цветовая схема должна включать следующие цвета:

- #1d3667 (синий);

- #FFFFFF(белый);

- #000000(черный).

Задание №2 Разработка алгоритмов и диаграмм

В ходе разработки был так же составлен словарь данных (Таблица 1 – 6).

Таблица 1 – Таблица «Сотрудники»

|  |  |  |  |
| --- | --- | --- | --- |
| Table name : Employee | | | |
| KEY | FIELD NAME | DATA TYPE / FIELD SIZE | REQUIRED? |
| PK | ID | INTEGER | N |
|  | LastName | VARCHAR (50) | N |
|  | FirstName | VARCHAR (50) | N |
|  | MiddleName | VARCHAR (50) | N |
|  | Cabinet | VARCHAR (10) | N |
|  | Address | VARCHAR (255) | N |
|  | Login | VARCHAR (50) | N |
|  | Password | VARCHAR (50) | N |
|  | RoleID | INTEGER | N |

Таблица 2 – Таблица «Роли»

|  |  |  |  |
| --- | --- | --- | --- |
| Table name : Role | | | |
| KEY | FIELD NAME | DATA TYPE / FIELD SIZE | REQUIRED? |
| PK | ID | INTEGER | N |
|  | RoleName | NVARCHAR (50) | N |

Таблица 3 – Таблица «Услуги»

|  |  |  |  |
| --- | --- | --- | --- |
| Table name : Service | | | |
| KEY | FIELD NAME | DATA TYPE / FIELD SIZE | REQUIRED? |
| PK | ID | INTEGER | N |
|  | ServiceName | VARCHAR (100) | N |
|  | DoctorID | INTEGER | N |
|  | Cost | DECIMAL(10, 2) | N |

Таблица 4 – Таблица «Записи на прием»

|  |  |  |  |
| --- | --- | --- | --- |
| Table name : Appointment | | | |
| KEY | FIELD NAME | DATA TYPE / FIELD SIZE | REQUIRED? |
| PK | ID | INTEGER | N |
|  | PatientID | INTEGER | N |
|  | ServiceID | INTEGER | N |
|  | AppointmentDateTime | DATETIME | N |

Таблица 5 – Таблица «Результаты»

|  |  |  |  |
| --- | --- | --- | --- |
| Table name : Result | | | |
| KEY | FIELD NAME | DATA TYPE / FIELD SIZE | REQUIRED? |
| PK | ID | INTEGER | N |
|  | AppointmentID | INTEGER | N |
|  | ResultDescription | TEXT | N |

Таблица 6 – Таблица «Пациенты»

|  |  |  |  |
| --- | --- | --- | --- |
| Table name : Patient | | | |
| KEY | FIELD NAME | DATA TYPE / FIELD SIZE | REQUIRED? |
| PK | ID | INTEGER | N |
|  | LastName | VARCHAR (50) | N |
|  | FirstName | VARCHAR (50) | N |
|  | MiddleName | VARCHAR (50) | N |
|  | BirthDate | DATE | N |
|  | Gender | VARCHAR (10) | N |
|  | Height | DECIMAL(5, 2) | N |
|  | Weight | DECIMAL(5, 2) | N |
|  | PassportDate | VARCHAR (20) | N |
|  | Email | VARCHAR (100) |  |

Далее были создана ER – модель(Рисунок 1) и база данных в Microsoft SQL Server, таблицы в которой были заполнены соответствующими данными (Рисунок 2 – 7).



Рисунок 1 – ER модель

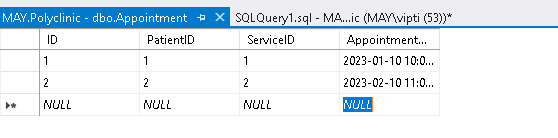


Рисунок 2 – Таблица «Записи на прием»

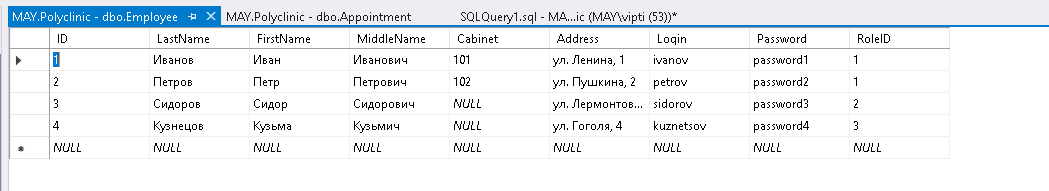


Рисунок 3 – Таблица «Сотрудники»

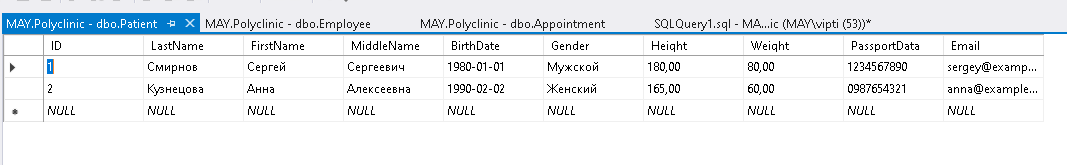


Рисунок 4 – Таблица «Пациенты»

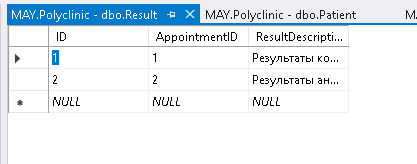


Рисунок 5 – Таблица «Результаты»

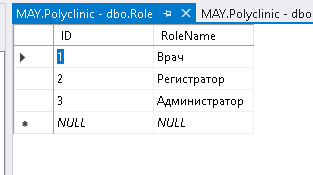


Рисунок 6 – Таблица «Роли»

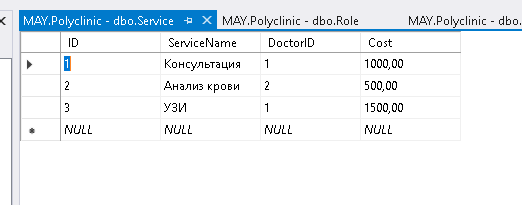


Рисунок 7 – Таблицы «Услуги»

Далее были созданы диаграммы прецендентов (Рисунок 8), последовательностей (Рисунок 9) и активности (Рисунок 10).



Рисунок 8 – Диаграмма прецендентов



Рисунок 9 – Диаграмма последовательностей



Рисунок 10 – Диаграмма активности

Задание №3 Разработка макетов программы

Следующим шагом было создание Wireframe макетов (Рисунок 11 – 12) для приложения и цветных макетов (Рисунок 13 – 14).

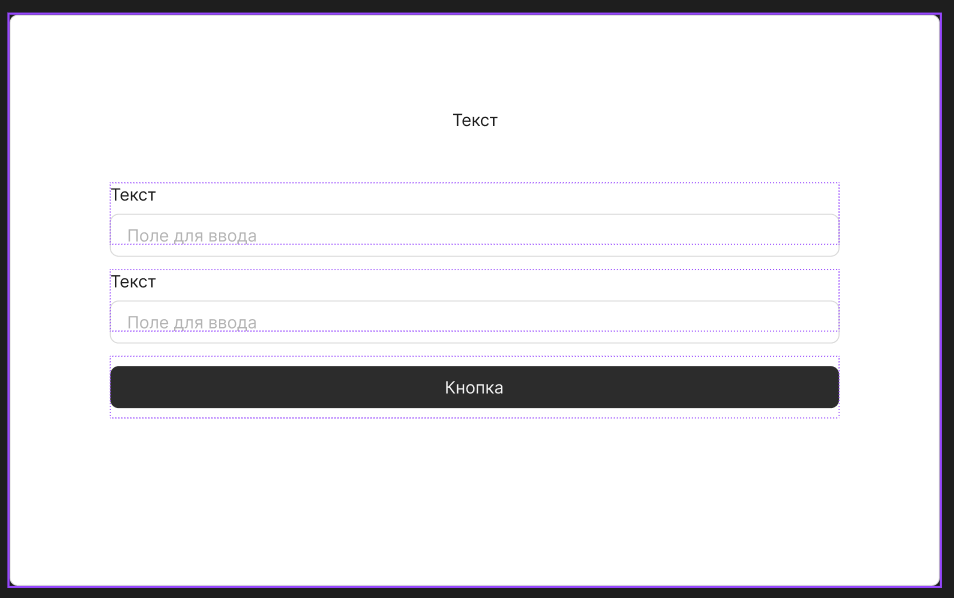


Рисунок 11 – Wireframe макет формы авторизации

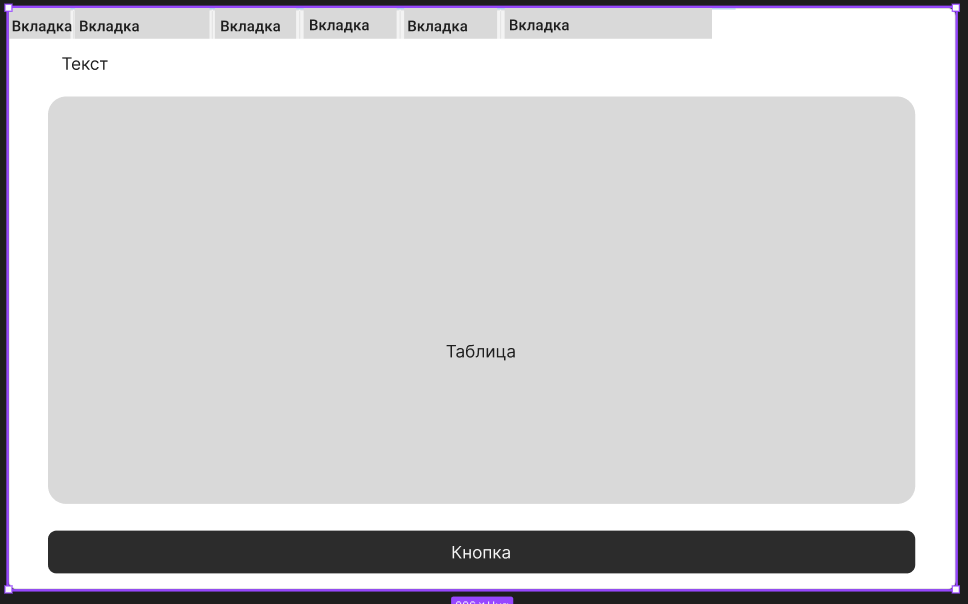


Рисунок 12 – Wireframe макет формы меню

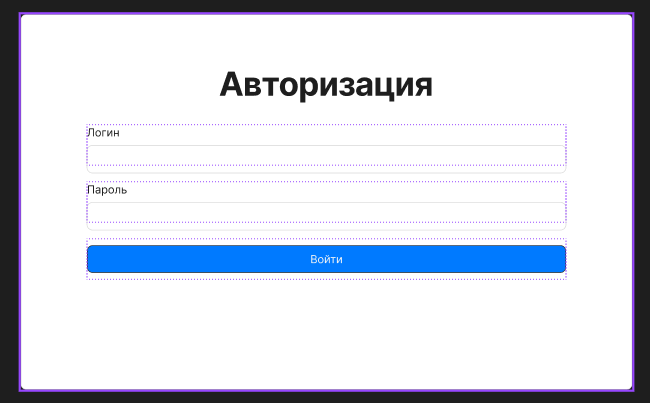


Рисунок 13 – Макет формы авторизации

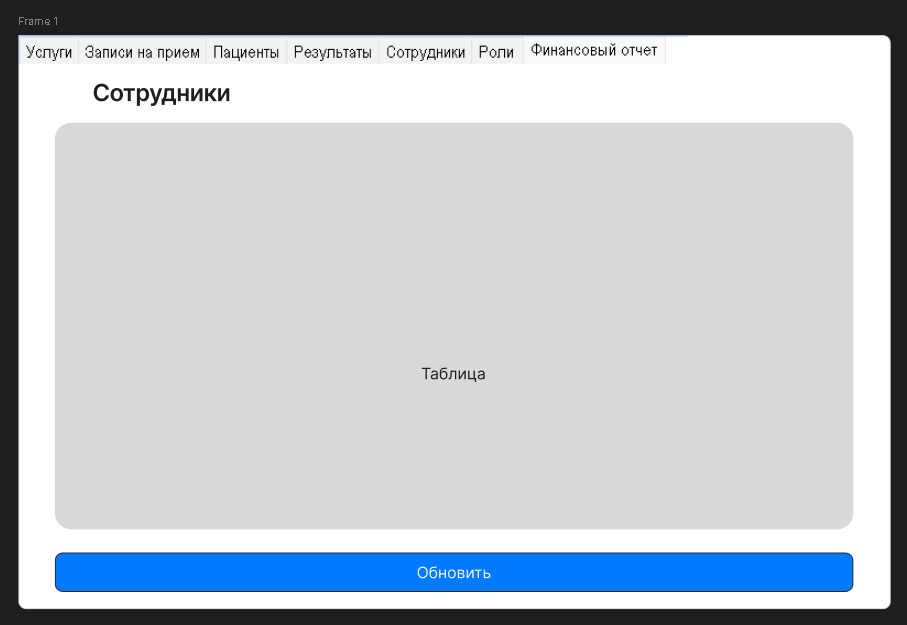


Рисунок 14 – Макет формы меню

После этого была создана карта навигации приложения (Рисунок 15).

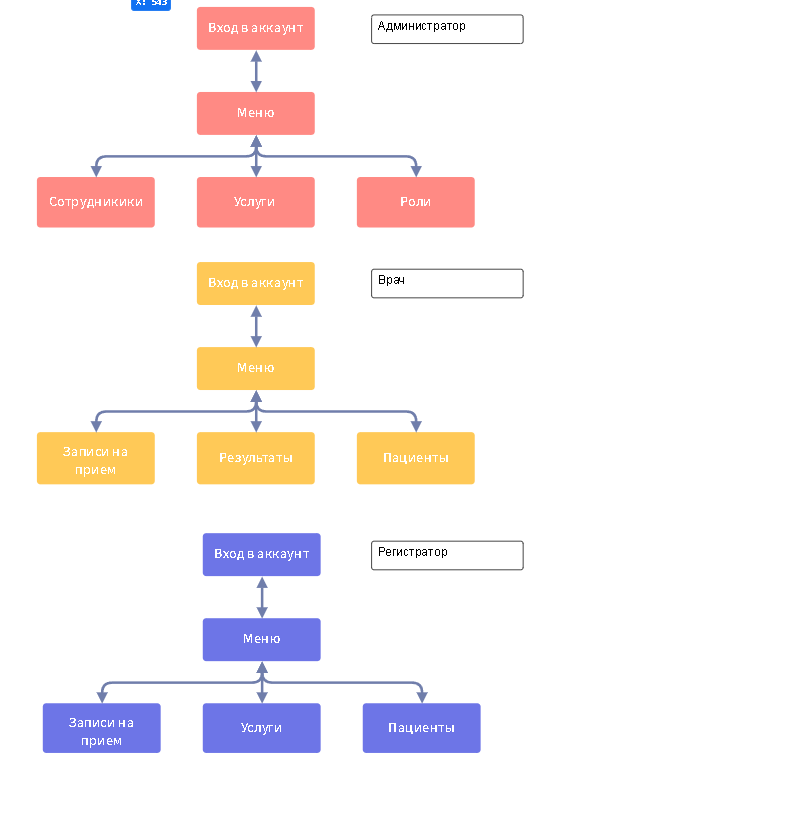


Рисунок 15 – Карта навигации приложения для разных ролей

Задание №4 Разработка программы

После создания макетов в Visual Studio Code (Рисунок 16 – 17), была произведена разработка функций для приложения. Код программы находится в приложении А.

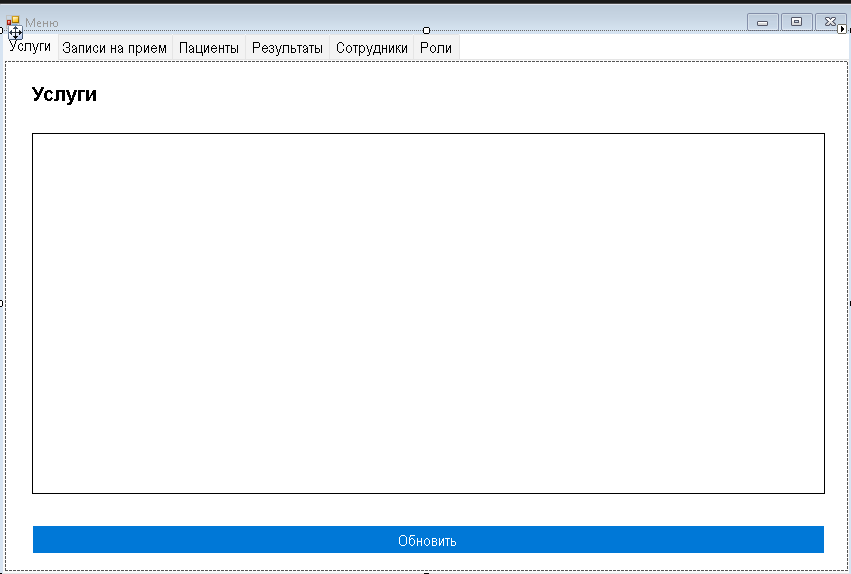


Рисунок 16 – Форма меню

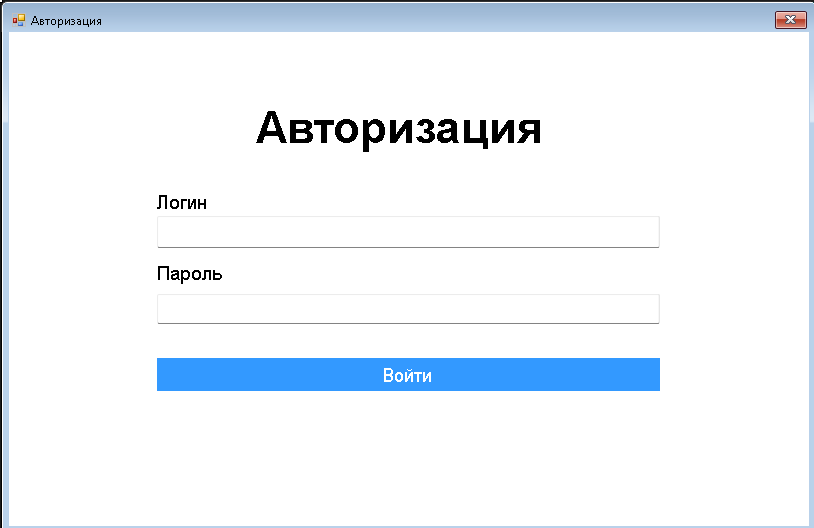


Рисунок 17 – Форма авторизации

Далее был произведен запуск приложения в роли администратора (Рисунок 18), в роли регистратора (Рисунок 19) и в роли врача (Рисунок 20).

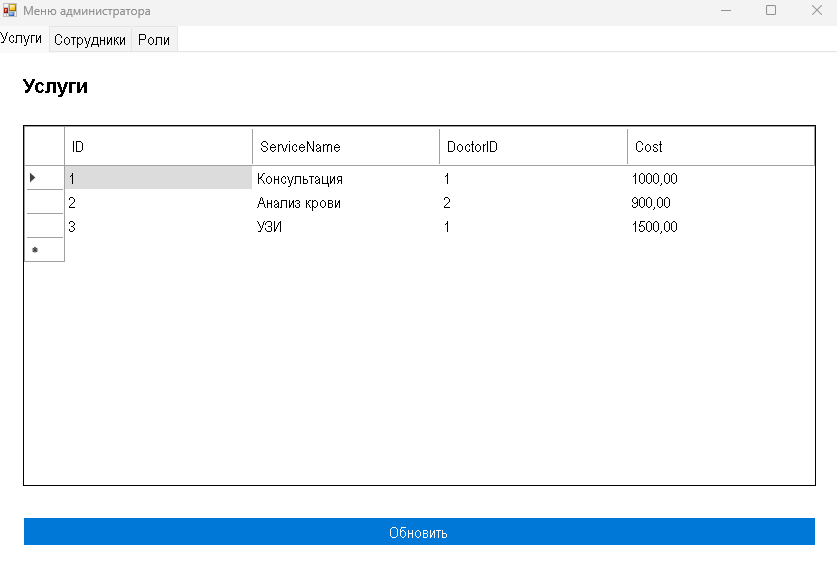


Рисунок 18 – Форма администратора для редактирования услуг

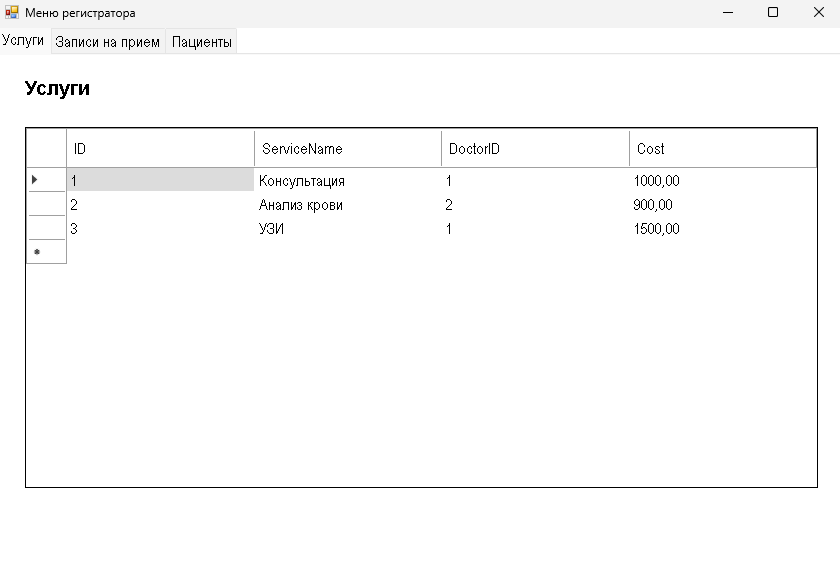


Рисунок 19 – Форма регистратора для просмотра услуг

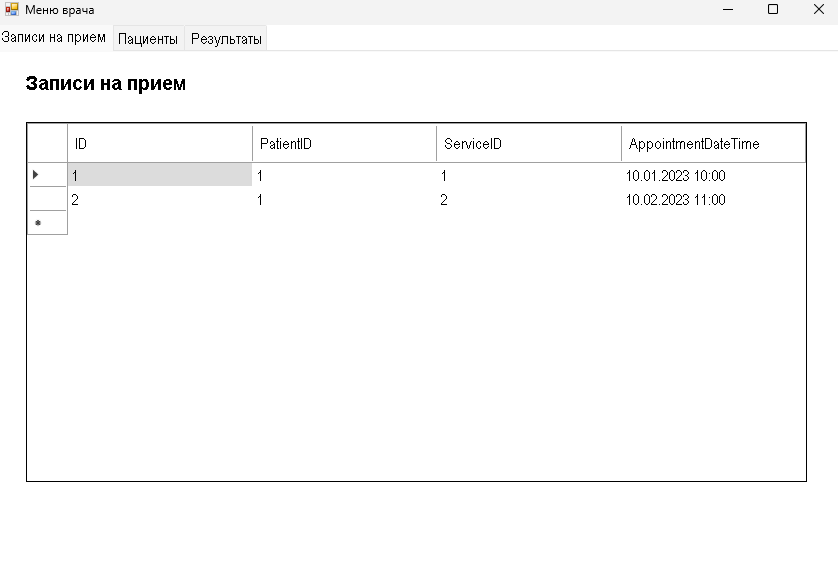


Рисунок 20 – Форма врача для просмотра записей на прием

Задание №5 Тестирование

Для произведения тестирования была создана аннотация теста

Таблица 7 – Аннотация теста

|  |  |
| --- | --- |
| Название проекта | Информационная система частной поликлиники |
| Рабочая версия | 1.0 |
| Имя тестирующего | Тина |
| Дата(ы) теста | 31.09.2024 – 01.10.2024 |

Таблица 8 – Test case #1

|  |  |
| --- | --- |
| Тестовый пример # | TC\_CH\_1 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Проверить доступ к изменению данных в таблице |
| Краткое изложение теста | Проверить, что пользователь может успешно изменять (Удалять/Добавлять/Редактировать) данные в таблице доступной ему для редактирования |
| Этапы теста | 1. Открыть приложение 2. Ввести действительный логин пользователя 3. Нажать кнопку "Войти" 4. Выбрать таблицу доступную для редактирования 5. Добавить запись в таблицу 6. Отредактировать запись в таблице 7. Удалить запись в таблице 8. Нажать на кнопку "Обновить" |
| Тестовые данные | * Логин пользователя: ivanov * Пароль: ivan * ID: 3 * AppointmentID: 1 * ResultDescription: “Результаты консультации: все показатели в норме.” |
| Ожидаемый результат | Пользователь успешно авторизуется и перенаправляется на главную страницу приложения, после чего редактирует (Изменяет/Добавляет/Удаляет) данные в таблице |
| Фактический результат | Пользователь успешно авторизуется и перенаправляется на главную страницу приложения, после чего редактирует (Изменяет/Добавляет/Удаляет) данные в таблице |
| Статус | Зачет |
| Предварительное условие | Пользователь должен быть зарегистрирован в системе |
| Постусловие | Пользователь должен быть авторизован и иметь доступ к функциям, соответствующим его роли, так же иметь доступ для редактирования доступных ему таблиц |
| Примечания/комментарии | Нет |

Таблица 9 – Test case #2

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_1 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверить доступ к таблицам для роли "Администратор" |
| Краткое изложение теста | Убедиться, что администратор имеет доступ к таблицам Роли, Сотрудники и Услуги |
| Этапы теста | 1. Авторизоваться как администратор 2. Попытаться открыть каждую таблицу 3. Попытаться редактировать данные в каждой таблице |
| Тестовые данные | * Логин пользователя: adm * Пароль: adm |
| Ожидаемый результат | Администратор имеет доступ к таблицам Сотрудники, Роли и Услуги и может редактировать данные в этих таблицах |
| Фактический результат | Администратор имеет доступ к таблицам Сотрудники, Роли и Услуги и может редактировать данные в этих таблицах |
| Статус | Зачет |
| Предварительное условие | Пользователь должен быть зарегистрирован как администратор |
| Постусловие | Администратор должен иметь доступ ко всем функциям редактирования для таблиц Сотрудники, Роли и Услуги |
| Примечания/комментарии | Нет |

Таблица 10 – Test case #3

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_2 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверить доступ к таблицам для роли "Врач" |
| Краткое изложение теста | Убедиться, что врач имеет доступ для редактирования таблицы Результаты и доступ для просмотра таблиц Пациенты и Записи на прием |
| Этапы теста | 1. Авторизоваться как врач 2. Попытаться открыть каждую таблицу 3. Попытаться редактировать данные в таблице Результаты 4. Попытаться редактировать таблицы Пациенты и Записи на прием |
| Тестовые данные | * Логин пользователя: petrov * Пароль: petr |
| Ожидаемый результат | Врач имеет доступ для редактирования таблицы Результаты и доступ для просмотра таблиц Записи на прием и Пациенты |
| Фактический результат | Врач имеет доступ для редактирования таблицы Результаты и доступ для просмотра таблиц Записи на прием и Пациенты |
| Статус | Зачет |
| Предварительное условие | Пользователь должен быть зарегистрирован как врач |
| Постусловие | Врач должен иметь доступ к функциям просмотра таблиц Записи на прием и Пациента и доступ для редактирования таблицы Результаты |
| Примечания/комментарии | Нет |

Таблица 11 – Test case #4

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_3 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверить доступ к таблицам для роли "Регистратор" |
| Краткое изложение теста | Убедиться, что регистратор имеет доступ для редактирования таблиц Пациенты и Записи на прием и доступ для просмотра таблицы Услуги |
| Этапы теста | 1. Авторизоваться как регистратор 2. Попытаться открыть каждую таблицу 3. Попытаться редактировать данные в таблицах Записи на прием и Пациенты 4. Попытаться редактировать данные в таблицах Услуги |
| Тестовые данные | * Логин пользователя: sidorov * Пароль: sidr |
| Ожидаемый результат | Регистратор имеет доступ для редактирования таблиц Записи на прием и Пациенты и доступ для просмотра таблицы Услуги |
| Фактический результат | Регистратор имеет доступ для редактирования таблиц Записи на прием и Пациенты и доступ для просмотра таблицы Услуги |
| Статус | Зачет |
| Предварительное условие | Пользователь должен быть зарегистрирован как регистратор |
| Постусловие | Регистратор должен иметь доступ для редактирования таблиц Пациенты и Записи на прием и доступ для просмотра таблицы Услуги |
| Примечания/комментарии | Нет |

Таблица 12 – Test case #5

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_4 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверить ошибку авторизации с недействительными учетными данными |
| Краткое изложение теста | Убедиться, что система выдает ошибку авторизации при попытке авторизации с недействительными учетными данными |
| Этапы теста | 1. Открыть приложение 2. Ввести недействительный логин пользователя 3. Ввести недействительный пароль пользователя 4. Нажать кнопку "Войти" |
| Тестовые данные | * Логин пользователя: kotov * Пароль: evgeny |
| Ожидаемый результат | Система выдает сообщение об ошибке "Неверный логин или пароль" |
| Фактический результат | Система выдает сообщение об ошибке "Неверный логин или пароль" |
| Статус | Зачет |
| Предварительное условие | Пользователь должен быть не зарегистрирован в системе |
| Постусловие | Пользователь не должен быть авторизован |
| Примечания/комментарии | Нет |

Далее были созданы 10 тестов для проверки работы приложения. Результат запуска всех тестов (Рисунок 21). Код Unit-tests находится в приложении Б.

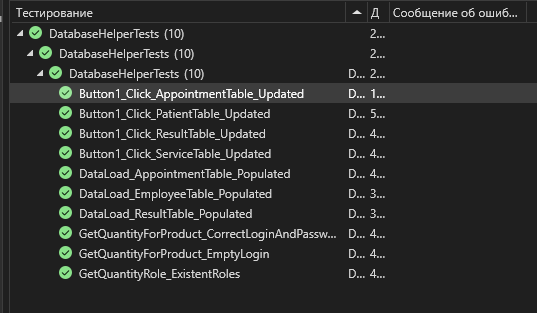


Рисунок 21 – Результат прохождения всех тестов

Задание №6 Git hub. Оценка проекта

Для сохранения данных проект был выгружен на Git Hub (Рисунок 22).

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики были успешно выполнены все поставленные задачи, начиная с анализа предметной области и заканчивая разработкой и тестированием программного продукта. Проведенный анализ методов решения и средств позволил выбрать оптимальные подходы и инструменты для реализации проекта. Исследовательские работы и разработка технического задания заложили прочную основу для дальнейшего проектирования и разработки.

Проектирование UML-диаграмм, включая диаграммы вариантов использования, последовательности, активности и состояния, позволило визуализировать и структурировать основные аспекты системы. Моделирование структуры программного обеспечения и разработка инфологической и даталогической моделей данных обеспечили надежную архитектуру и эффективное управление данными.

Проектирование интерфейса пользователя и разработка алгоритмов программного продукта были направлены на создание удобного и функционального приложения. Создание и тестирование программного продукта, включая модульное тестирование и отладку, позволили выявить и устранить возможные ошибки, обеспечив высокое качество конечного продукта.

В результате выполненной практики были получены ценные знания и навыки, которые будут полезны в дальнейшей профессиональной деятельности. Практика также подтвердила важность системного подхода и внимания к деталям на всех этапах разработки программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Поликлиника

{

public partial class Form1 : Form

{

private SqlDataAdapter adapter;

private DataTable table;

private string connectionString = "Data Source = MAY; Initial Catalog = Polyclinic; Integrated Security = True";

public Form1()

{

InitializeComponent();

adapter = new SqlDataAdapter();

table = new DataTable();

}

private void button1\_Click(object sender, EventArgs e)

{

string querystring = $"select Employee.ID, Login, Password, RoleName from Employee, Role where RoleID = Role.ID and Login ='{textBox1.Text}' and Password= '{textBox2.Text}' ";

using (SqlConnection connection = new SqlConnection(connectionString))

{

adapter = new SqlDataAdapter(querystring, connection);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

table = new DataTable();

adapter.Fill(table);

}

if (table.Rows.Count == 1)//если запрос вернул одну строку с найденным пользователем

{

Menu form2 = new Menu(table.Rows[0].ItemArray[3].ToString());

this.Hide();

form2.ShowDialog();

this.Show();

textBox1.Clear();

textBox1.Focus();

textBox2.Clear();

}

else MessageBox.Show("Логин или пароль введены неверно", "Аккаут не существует!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Enter && sender == textBox1)

{

e.KeyChar = '\0';

textBox2.Focus();

}

else if (e.KeyChar == (char)Keys.Enter && sender == textBox2)

{

e.KeyChar = '\0';

button1.Focus();

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text.Length > 0 && textBox2.Text.Length > 0) button1.Enabled = true;

else button1.Enabled = false;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics.Eventing.Reader;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Поликлиника

{

public partial class Menu : Form

{

string connectionString = "Data Source = MAY; Initial Catalog = Polyclinic; Integrated Security = True";

private SqlDataAdapter dataAdapterService = new SqlDataAdapter();

private BindingSource bindingSourceService = new BindingSource();

private SqlDataAdapter dataAdapterAppointment = new SqlDataAdapter();

private BindingSource bindingSourceAppointment = new BindingSource();

private SqlDataAdapter dataAdapterPatient = new SqlDataAdapter();

private BindingSource bindingSourcePatient = new BindingSource();

private SqlDataAdapter dataAdapterResult = new SqlDataAdapter();

private BindingSource bindingSourceResult = new BindingSource();

private SqlDataAdapter dataAdapterEmployee = new SqlDataAdapter();

private BindingSource bindingSourceEmployee = new BindingSource();

private SqlDataAdapter dataAdapterRole = new SqlDataAdapter();

private BindingSource bindingSourceRole = new BindingSource();

public Menu(string status)

{

InitializeComponent();

if (status == "Администратор")

{

tabPage2.Parent = null;

tabPage3.Parent = null;

tabPage4.Parent = null;

}

if (status == "Врач")//234

{

tabPage1.Parent = null;

tabPage5.Parent = null;

tabPage6.Parent = null;

dataGridView6.ReadOnly = true;

dataGridView5.ReadOnly = true;

}

if (status == "Регистратор")//123

{

tabPage4.Parent = null;

tabPage5.Parent = null;

tabPage6.Parent = null;

dataGridView7.ReadOnly = true;

}

string q1 = "SELECT \* FROM Service";

DataLoad(ref dataAdapterService, bindingSourceService, q1, dataGridView7);

string q2 = "SELECT \* FROM Appointment";

DataLoad(ref dataAdapterAppointment, bindingSourceAppointment, q2, dataGridView6);

string q3 = "SELECT \* FROM Patient";

DataLoad(ref dataAdapterPatient, bindingSourcePatient, q3, dataGridView5);

string q4 = "SELECT \* FROM Result";

DataLoad(ref dataAdapterResult, bindingSourceResult, q4, dataGridView4);

string q5 = "SELECT \* FROM Employee";

DataLoad(ref dataAdapterEmployee, bindingSourceEmployee, q5, dataGridView3);

string q6 = "SELECT \* FROM Role";

DataLoad(ref dataAdapterRole, bindingSourceRole, q6, dataGridView2);

}

private void DataLoad(ref SqlDataAdapter dataAdapter, BindingSource bindingSource, string q, DataGridView dataGridView)

{

dataAdapter = new SqlDataAdapter(q, connectionString);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);

DataTable table = new DataTable();

dataAdapter.Fill(table);

bindingSource.DataSource = table;

dataGridView.DataSource = bindingSource;

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

dataAdapterService.Update((DataTable)bindingSourceService.DataSource);

}catch (Exception ex)

{

MessageBox.Show("Ошибка: " + ex.Message, "Ошибка");

}

}

}

}

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Поликлиника;

ПРИЛОЖЕНИЕ Б

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Data;

using NUnit.Framework;

using System;

namespace DatabaseHelperTests

{

[TestFixture]

public class DatabaseHelperTests

{

private string connectionString = "Data Source = MAY; Initial Catalog = Polyclinic; Integrated Security = True";

[Test]

public void GetQuantityRole\_ExistentRoles()

{

// Подготовка теста: создание запроса для получения всех ролей

string querystring = $"select Role.ID from Role";

SqlDataAdapter adapter = new SqlDataAdapter(querystring, connectionString);

// Выполнение теста: заполнение DataTable данными из базы данных

DataTable table = new DataTable();

adapter.Fill(table);

// Проверка результата: ожидается, что таблица будет содержать 3 строки (3 роли)

Microsoft.VisualStudio.TestTools.UnitTesting.Assert.AreEqual(3, table.Rows.Count);

}

[Test]

public void GetQuantityForProduct\_EmptyLogin()

{

// Подготовка теста: создание запроса с пустым логином

string querystring = $"select Employee.ID, Login, Password, RoleName from Employee, Role where RoleID = Role.ID and Login ='' and Password= 'password' ";

SqlDataAdapter adapter = new SqlDataAdapter(querystring, connectionString);

// Выполнение теста: заполнение DataTable данными из базы данных

DataTable table = new DataTable();

adapter.Fill(table);

// Проверка результата: ожидается, что таблица будет пустой (0 строк)

Microsoft.VisualStudio.TestTools.UnitTesting.Assert.AreEqual(0, table.Rows.Count);

}

[Test]

public void GetQuantityForProduct\_CorrectLoginAndPassword()

{

// Подготовка теста: создание запроса с корректными логином и паролем

string querystring = $"select Employee.ID, Login, Password, RoleName from Employee, Role where RoleID = Role.ID and Login ='adm' and Password= 'adm' ";

SqlDataAdapter adapter = new SqlDataAdapter(querystring, connectionString);

// Выполнение теста: заполнение DataTable данными из базы данных

DataTable table = new DataTable();

adapter.Fill(table);

// Проверка результата: ожидается, что таблица будет содержать одну строку (1 пользователь)

Microsoft.VisualStudio.TestTools.UnitTesting.Assert.AreEqual(1, table.Rows.Count);

}

private Menuu \_menuu;

[SetUp]

public void SetUp()

{

// Инициализация объекта Menuu перед выполнением тестов

\_menuu = new Menuu("Администратор");

}

[Test]

public void DataLoad\_AppointmentTable\_Populated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Appointment

string q = "SELECT \* FROM Appointment";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

// Выполнение теста: загрузка данных в DataGridView

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Проверка результата: ожидается, что данные будут загружены и таблица будет содержать строки

NUnit.Framework.Assert.That(bindingSource.DataSource, Is.Not.Null);

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

[Test]

public void DataLoad\_ResultTable\_Populated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Result

string q = "SELECT \* FROM Result";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

// Выполнение теста: загрузка данных в DataGridView

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Проверка результата: ожидается, что данные будут загружены и таблица будет содержать строки

NUnit.Framework.Assert.That(bindingSource.DataSource, Is.Not.Null);

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

[Test]

public void DataLoad\_EmployeeTable\_Populated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Employee

string q = "SELECT \* FROM Employee";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

// Выполнение теста: загрузка данных в DataGridView

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Проверка результата: ожидается, что данные будут загружены и таблица будет содержать строки

NUnit.Framework.Assert.That(bindingSource.DataSource, Is.Not.Null);

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

[Test]

public void Button1\_Click\_ServiceTable\_Updated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Service

string q = "SELECT \* FROM Service";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Выполнение теста: нажатие на кнопку для обновления данных

\_menuu.button1\_Click(null, null);

// Проверка результата: ожидается, что данные будут обновлены и таблица будет содержать строки

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

[Test]

public void Button1\_Click\_AppointmentTable\_Updated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Appointment

string q = "SELECT \* FROM Appointment";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Выполнение теста: нажатие на кнопку для обновления данных

\_menuu.button1\_Click(null, null);

// Проверка результата: ожидается, что данные будут обновлены и таблица будет содержать строки

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

[Test]

public void Button1\_Click\_PatientTable\_Updated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Patient

string q = "SELECT \* FROM Patient";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Выполнение теста: нажатие на кнопку для обновления данных

\_menuu.button1\_Click(null, null);

// Проверка результата: ожидается, что данные будут обновлены и таблица будет содержать строки

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

[Test]

public void Button1\_Click\_ResultTable\_Updated()

{

// Подготовка теста: создание запроса для получения данных из таблицы Result

string q = "SELECT \* FROM Result";

SqlDataAdapter dataAdapter = new SqlDataAdapter(q, \_menuu.connectionString);

BindingSource bindingSource = new BindingSource();

DataGridView dataGridView = new DataGridView();

\_menuu.DataLoad(ref dataAdapter, bindingSource, q, dataGridView);

// Выполнение теста: нажатие на кнопку для обновления данных

\_menuu.button1\_Click(null, null);

// Проверка результата: ожидается, что данные будут обновлены и таблица будет содержать строки

NUnit.Framework.Assert.That(((DataTable)bindingSource.DataSource).Rows.Count, Is.GreaterThan(0));

}

}

}

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Дуларидзе Тынатын Тамазовна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «05» октября 2024 г.