Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Конструирование программного обеспечения

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**СИСТЕМА УПРАВЛЕНИЯ ЛАБОРАТОРНЫМИ РАБОТАМИ И ОТЧЁТАМИ ПО ПРЕДМЕТУ СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ**

БГУИР КР 6 - 05 - 06 12 01 029 ПЗ

Выполнил: студент группы 310901 Усов А. М.

Проверил: Василькова А. Н.

Минск 2024

Содержание

[Введение 4](#_Toc900729803)

[1 Постановка задачи 5](#_Toc1907954399)

[1.1 Описание предметной области 5](#_Toc1215652243)

[1.2 Сравнительный анализ существующих решений 6](#_Toc204304474)

[1.3 Информационная база задачи 10](#_Toc1260996811)

[1.4 Функциональное назначение 11](#_Toc1407870498)

[2 Проектирование задачи 12](#_Toc171933912)

[2.1 Алгоритм решения задачи 12](#_Toc818830875)

[2.2 Логическое моделирование 12](#_Toc1209878580)

[2.3 Выбор и обоснование инструментов разработки 13](#_Toc1287252116)

[3 Программная реализация 15](#_Toc877094095)

[3.1 Физическая структура 15](#_Toc930106346)

[3.2 Описание разработанных модулей 16](#_Toc373395914)

[4 Тестирование 19](#_Toc758714259)

[5 Применение программы 23](#_Toc1235220066)

[5.1 Руководство пользователя 23](#_Toc1046107490)

[Приложение А 29](#_Toc772501982)

Введение

В 21 веке все этапы взаимодействия между людьми становятся удаленными от привычного общения и приобретают онлайн характер. В связи с этим, есть необходимость с улучшением взаимодействия людей, в частности преподавателя со студентами посредством модернизации процесса взаимодействия путем переноса в информационные сети.

На данный момент, если студент хочет сдать преподавателю выполненную лабораторную работу, например по дисциплине современные языки программирования, ему необходимо весь исполняемый код показать преподавателю лично, что занимает гораздо больше времени, нежели если бы студенты отправляли готовые лабораторные работы на проверку по сети с использованием автоматизированных систем с широким функционалом.

В мире уже есть готовые решения, которые позволяют проверять написанный человеком код другим человеком, и оставлять последующие комментарии для более точной оценки. Но в данных решениях достаточно сложно разобраться, если до этого никогда не взаимодействовали с такими системами, что может оттолкнуть пользователя от упрощения будущего взаимодействия и не видеть такого пути оптимизации.

Цель данной курсовой работы – разработать и реализовать программное средство «Система управления лабораторными работами и отчётами по предмету современные языки программирования» для упрощения взаимодействия преподавателя со студентами в контексте проверки лабораторных работ.

Для достижения поставленной цели были сформулированы следующие задачи:

* провести анализ требований у преподавателей и студентов про необходимую функциональность;
* изучить существующие программные средства, выявить из достоинства и недостатки;
* спроектировать архитектуру приложения;
* реализовать клиент-серверное взаимодействие;
* реализовать функционал, необходимый для получения и сдачи лабораторных работ через программное средство;
* реализовать простой и понятный интерфейс;
* система должна быть легка в освоении и ориентирована на учебные заведения.

1 Постановка задачи

1.1 Описание предметной области

Система управления лабораторными работами по дисциплине «Современные языки программирования» предназначена для автоматизации процесса взаимодействия между преподавателем и студентами в рамках учебного процесса. Основная цель системы – упрощение процесса сдачи лабораторных работ, проверки и хранения результатов, а также улучшение коммуникации между участниками образовательного процесса.

Основными пользователями системы являются преподаватели и студенты. Для каждого из них предусмотрены свои роли и функции, соответствующие их обязанностям в учебном процессе.

Преподаватели осуществляют следующие основные операции:

* создание лабораторных заданий: Преподаватель формирует задания по лабораторным работам, задаёт сроки их выполнения, а также прикрепляет необходимые материалы и инструкции. Эти данные автоматически становятся доступными студентам;
* приём и проверка работ: после сдачи лабораторных работ студентами, преподаватель получает доступ к файлам для проверки, выставляет оценки и оставляет комментарии с замечаниями. Это позволяет быстро оценить результаты и дать обратную связь;

Студенты выполняют следующие основные операции:

* получение и изучение заданий: Студенты получают доступ к лабораторным работам, могут ознакомиться с требованиями, сроками и дополнительными материалами. Это позволяет им планировать время и ресурсы для выполнения работы;
* сдача отчётов: после выполнения задания студент загружает отчёт через систему, что фиксирует дату сдачи и позволяет отслеживать, какие работы уже выполнены;
* просмотр результатов: после проверки работы студент может ознакомиться с выставленной оценкой и комментариями преподавателя.

С точки зрения автоматизации данных процессов, система должна:

1. Поддерживать актуальную базу данных лабораторных заданий с указанием сроков выполнения, инструкций и требований.
2. Обеспечивать простой и удобный интерфейс для сдачи отчётов, проверки и комментирования работ.
3. Поддерживать коммуникацию между преподавателями и студентами посредством обмена сообщениями и уведомлениями о результатах и сроках.

Автоматизация процесса сдачи лабораторных работ и их проверки позволит сократить время на выполнение рутинных операций, снизить вероятность ошибок при проверке и улучшить качество взаимодействия между преподавателем и студентом.

1.2 Сравнительный анализ существующих решений

Для разработки системы управления лабораторными работами и отчётами по предмету «Современные языки программирования» важно рассмотреть существующие решения и проанализировать их функциональные особенности. Это поможет выявить сильные и слабые стороны аналогичных систем и определить ключевые требования для проектируемой системы.

На данный момент существует несколько типов систем для автоматизации учебного процесса, включая системы управления учебными курсами (*Learning Management Systems — LMS*) и специализированные платформы для проверки программных проектов. Рассмотрим наиболее популярные из них:

**1 *Moodle*.**

*Moodle* – это одна из самых популярных систем управления учебным процессом, используемая во многих образовательных учреждениях. Она предоставляет широкий набор функций для преподавателей и студентов:

* создание и управление курсами, тестами и заданиями;
* возможность сдачи лабораторных работ в электронном виде;
* проверка работ преподавателями и автоматическая проверка тестов;
* мощные средства для отслеживания прогресса студентов.

Преимущества:

* полный набор инструментов для управления курсами;
* гибкость в настройке под нужды конкретного учебного заведения;
* активное сообщество разработчиков и пользователей.

Недостатки:

* сложность настройки и управления для преподавателей;
* избыточность для небольших курсов, что может усложнять процесс.

Главная страница сервиса представлена на рисунке 1.1

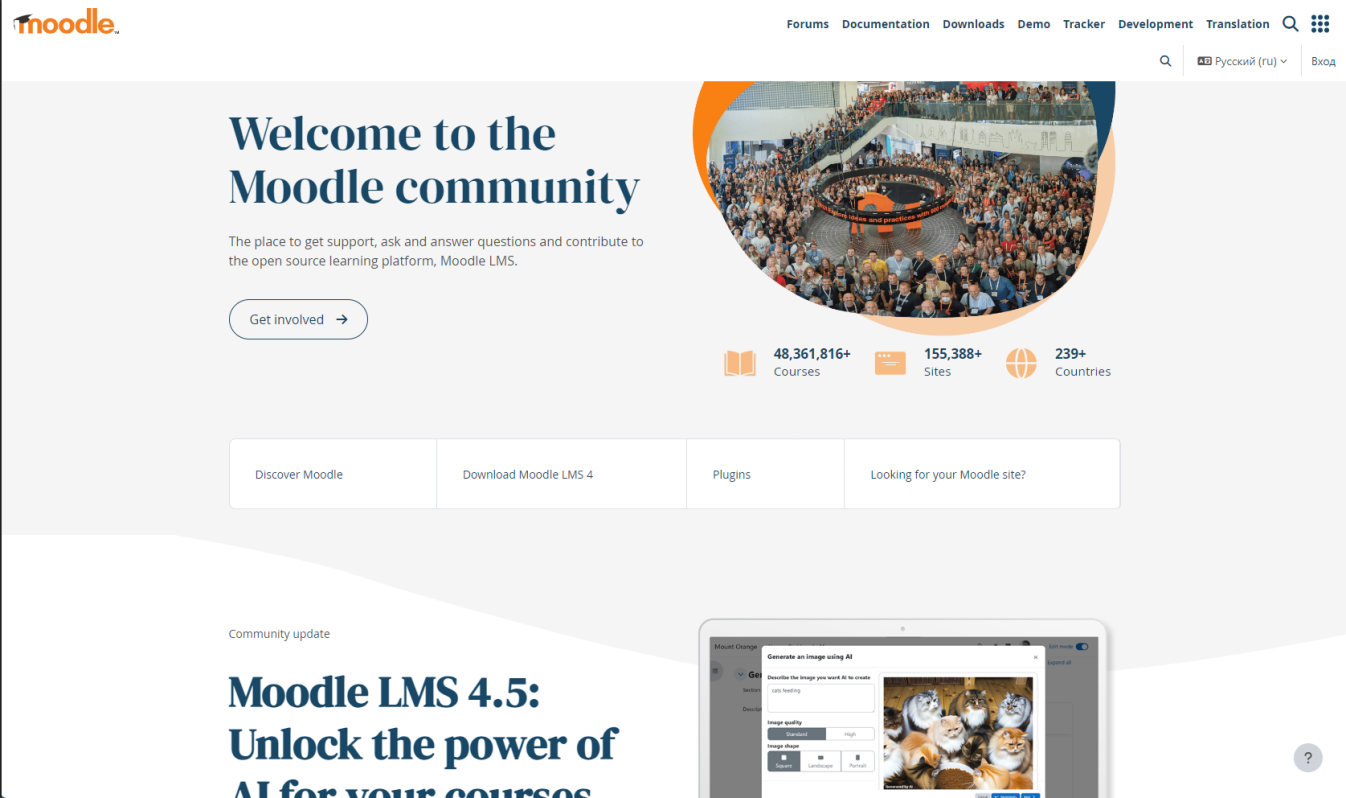


Рисунок 1.1 – Главная страница сервиса ***Moodle***

**2 *Google* *Classroom*.**

*Google* *Classroom* представляет собой простую и интуитивно понятную платформу для организации учебных процессов:

* простота в использовании и интеграция с другими сервисами *google*;
* лёгкий обмен заданиями, отчётами и комментариями;
* возможность сдачи отчётов по лабораторным работам.

Преимущества:

* простота использования, не требует особых технических навыков;
* бесплатность и интеграция с другими сервисами *google*.

Недостатки:

* ограниченный функционал для сложных курсов, связанных с программированием;
* отсутствие инструментов для автоматической проверки программного кода.

Главная страница сервиса представлена на рисунке 1.2.

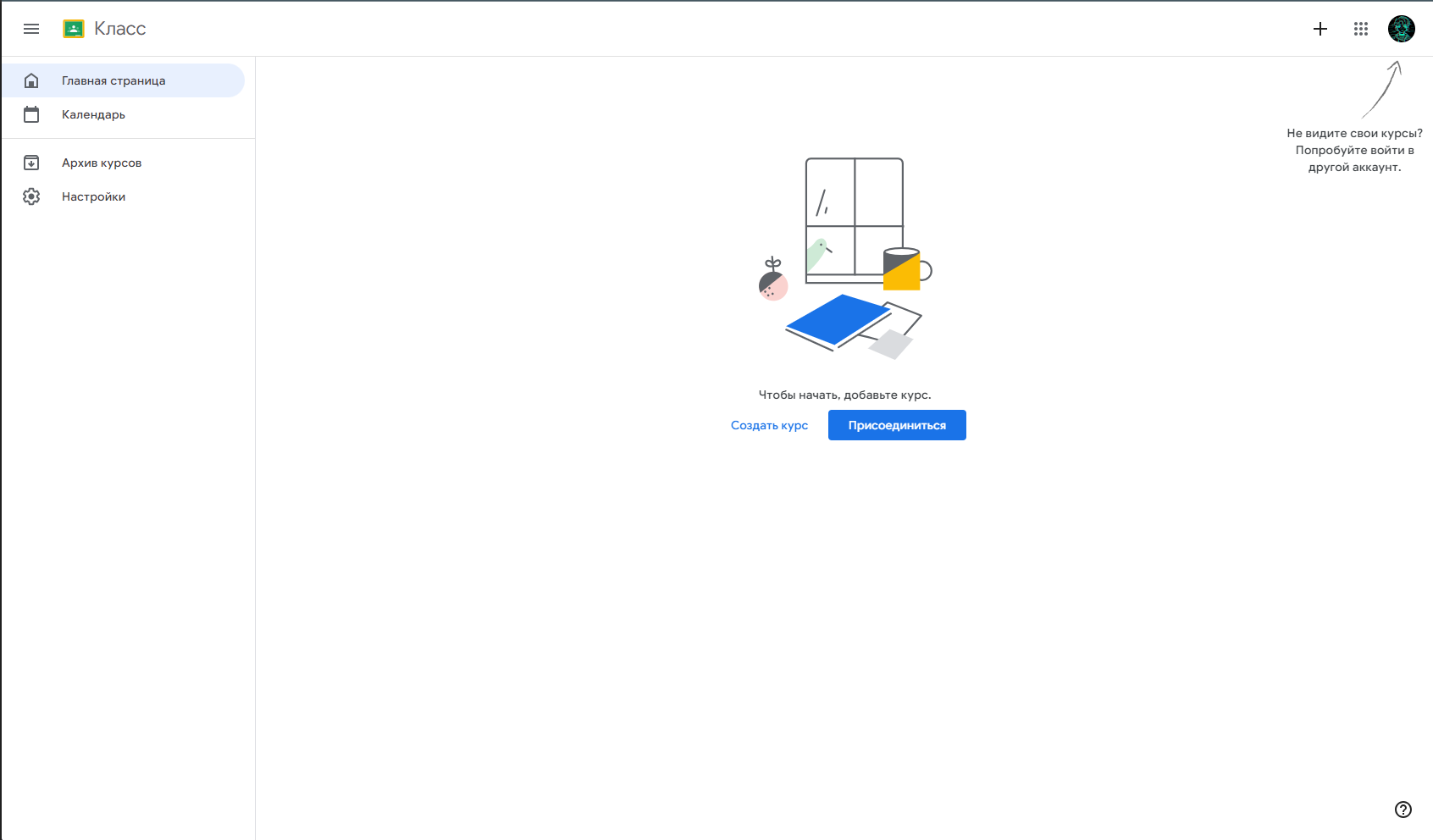


Рисунок 1.2 – Главная страница сервиса *Google Classroom*

**3 *Codeforces* и *LeetCode*.**

Специализированные платформы для выполнения и проверки программных заданий:

* ориентированы на сдачу и автоматическую проверку программного кода;
* поддержка широкого спектра языков программирования;
* встроенная система рейтингов и соревнований.

Преимущества:

* мгновенная проверка программного кода;
* интерактивное обучение и соревнования.

Недостатки:

* не поддерживают все аспекты учебного процесса, такие как сдача отчётов и обратная связь преподавателей;
* не подходят для курсов с широким теоретическим содержанием.

Главнае страницы сервисов представлены на рисунках 1.3–1.4.

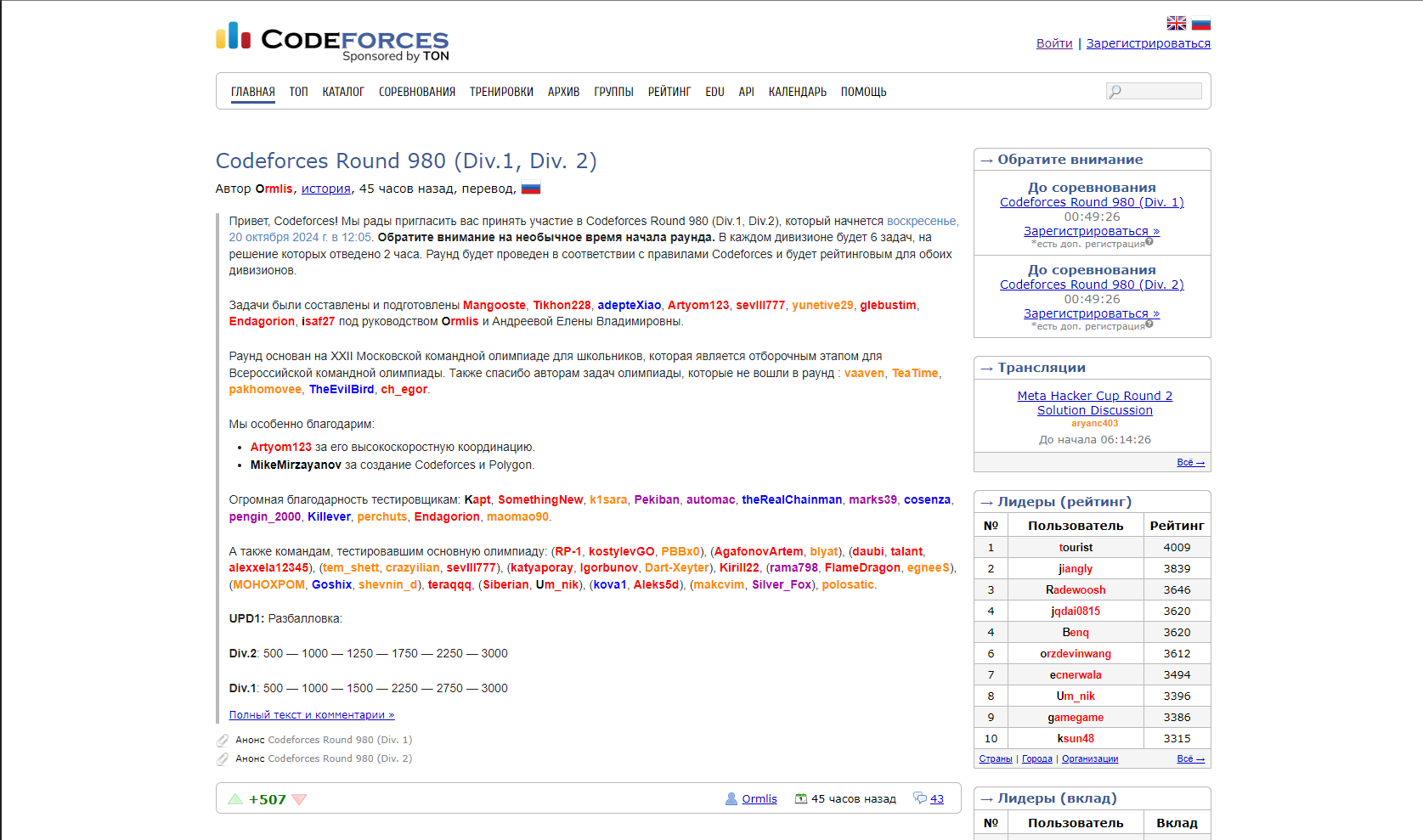


Рисунок 1.3 – Главная страница сервиса *Codeforces*

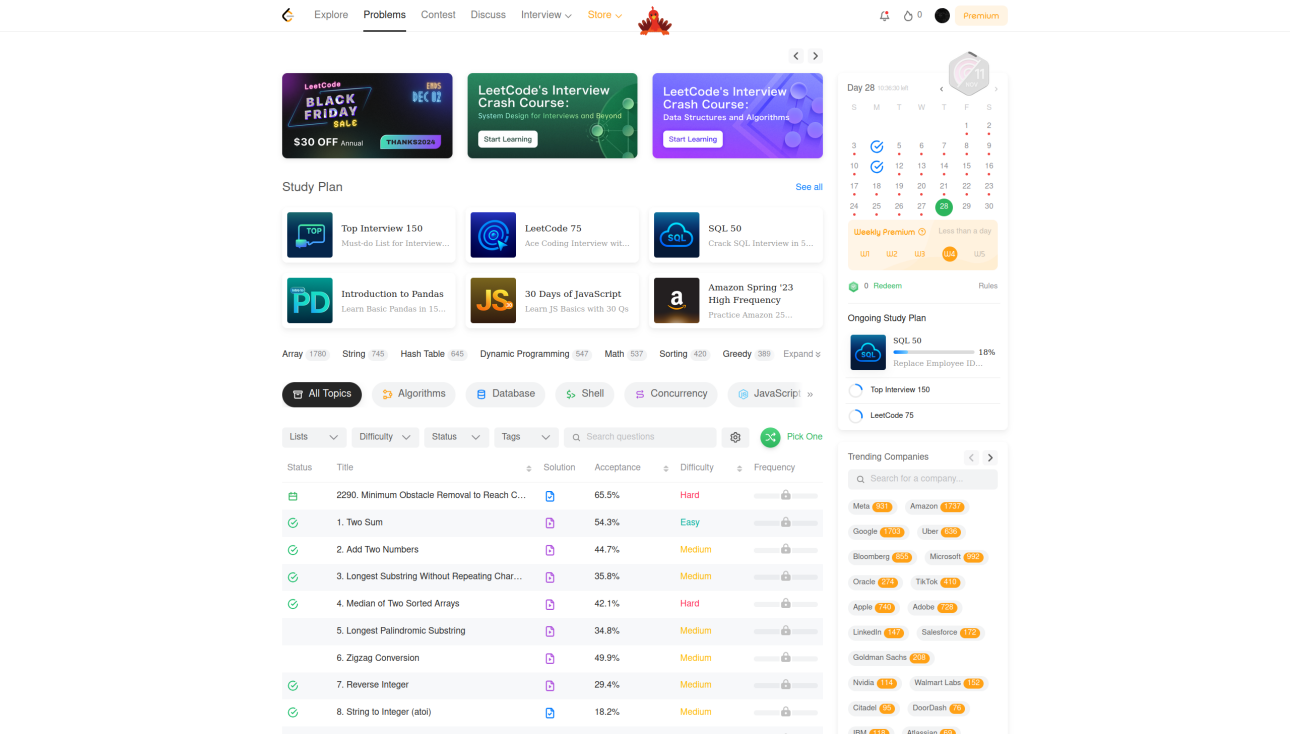


Рисунок 1.4 – Главная страница сервиса *LeetCode*

Все представленные решения обладают достоинствами и могут быть полезны в различных контекстах, однако они имеют свои ограничения. *Moodle* и *Google Classroom* удобны для организации общего учебного процесса, но не всегда хорошо поддерживают автоматизацию сдачи и проверки программных проектов. *Codeforces* и *LeetCode*, напротив, идеально подходят для практической проверки навыков программирования, однако ограничены в плане управления теоретическими заданиями и лабораторными отчётами.

Таким образом, проектируемая система управления лабораторными работами и отчётами должна сочетать лучшие черты существующих решений:

* Простоту использования и управления (как в *Google Classroom*);
* Гибкость в настройке лабораторных заданий и оценок (как в *Moodle*).

Это позволит создать более эффективный инструмент для преподавателей и студентов, ориентированный на конкретные требования курса по современным языкам программирования.

1.3 Информационная база задачи

Информационная база задачи представляет собой набор данных, необходимых для эффективного управления лабораторными работами и отчётами по предмету «Современные языки программирования». Система обеспечивает автоматизацию процессов загрузки, проверки и хранения лабораторных работ, а также упрощает взаимодействие между студентами и преподавателями.

Основные элементы информационной базы включают:

* лабораторные работы: система должна содержать информацию о всех лабораторных заданиях, включая их темы, требования, сроки сдачи и материалы. Для каждой лабораторной работы фиксируются данные о её создании, статусе (выполняется, сдана, проверена) и оценке;
* студенты: для каждого студента создаётся личный кабинет, в котором хранятся персональные данные, список лабораторных работ и их статусы. Студенты могут загружать выполненные задания и просматривать результаты проверки. Также фиксируется информация о сроках сдачи;
* преподаватели: в системе хранятся данные о преподавателях, которые могут создавать, редактировать и удалять лабораторные задания, проверять сданные работы студентов, выставлять оценки и оставлять комментарии. Преподаватели также могут отслеживать прогресс выполнения лабораторных работ всеми студентами;
* отчёты: для каждой лабораторной работы создаётся отчёт, включающий выполненные задания, код программ и результаты их выполнения. Эти отчёты доступны для преподавателей для проверки и анализа, а также для студентов. Для каждого отчёта сохраняется уникальный идентификатор, данные о студенте и преподавателе, а также информация о результатах проверки;
* система оценки и статусов: каждая лабораторная работа проходит через определённые стадии: от создания задания и до его проверки и оценки. Статусы работ обновляются в зависимости от действий студентов и преподавателей;
* права доступа: для корректной работы системы устанавливаются разные уровни доступа для студентов и преподавателей. Студенты имеют доступ к своим заданиям и отчётам, а преподаватели – к заданиям всех студентов, с возможностью их проверки и оценки;

Информационная база реализуется на основе *SQL* базы данных, что позволяет удобно хранить лабораторные работы и отчёты, обеспечивая их целостность и актуальность. Система автоматически обновляет данные при загрузке новых работ или после их проверки преподавателем, что исключает возможность ошибок и устаревания информации.

1.4 Функциональное назначение

Функциональное назначение системы управления лабораторными работами и отчётами по дисциплине «Современные языки программирования» заключается в автоматизации процессов, связанных с созданием, выполнением, сдачей и проверкой лабораторных работ. Основные функции системы направлены на упрощение взаимодействия между студентами и преподавателями и сокращение временных затрат на выполнение рутинных операций.

Основные функции системы включают:

* создание и управление лабораторными заданиями: преподаватели могут создавать лабораторные работы, добавлять необходимые материалы и устанавливать сроки сдачи. Система позволяет редактировать задания и управлять их доступностью для студентов.
* загрузка и сдача лабораторных работ: студенты могут загружать выполненные работы через личный кабинет, где фиксируется дата и время загрузки. Система автоматически проверяет наличие опозданий, учитывая установленные дедлайны, и предоставляет преподавателям доступ к сданным работам для их проверки;
* проверка и оценивание: преподаватели имеют возможность проверять загруженные студентами работы, выставлять оценки и оставлять комментарии. Система сохраняет результаты проверки и автоматически обновляет статус лабораторной работы для каждого студента (в очереди, проверено);
* информационная поддержка студентов: в системе реализован функционал, позволяющий студентам получать доступ к информации о каждой лабораторной работе.

Более наглядно функционал программы представлен на диаграмме вариантов использования в графической части на листе 1.

Функционал системы должен быть интуитивно понятным и лёгким в использовании как для преподавателей, так и для студентов. Автоматизация всех ключевых процессов позволяет сократить время на выполнение рутинных операций, минимизировать ошибки и сделать учебный процесс более прозрачным и управляемым.

2 Проектирование задачи

2.1 Алгоритм решения задачи

Алгоритм решения задачи для системы управления лабораторными работами и отчётами по дисциплине современные языки программирования представляет собой последовательность действий и алгоритмов, необходимых для решения задач, таких как:

1. Безопасность данных предполагает собой наличие шифрования данных. Пароли аккаунтов администраторов и кассиров будут храниться в хешированном виде. Для хеширования будет применяться криптографический алгоритм *SHA-256*. Для проверки введенного пароля, он будет также захеширован, а затем сверен с существующим хешем пароля. У двух идентичных паролей хеши совпадают.
2. Будет разработан удобный, интуитивно понятный и адаптивный интерфейс для комфортной работы пользователя. Для упрощения работы с большим объемом данных будут разработаны алгоритмы упорядочивания (сортировки).
3. Вся информация будет храниться в базе даннчх, что обеспечивает простую работу с данными. Будут реализованы алгоритмы сбора существующей и запись новой информации в базу данных.
4. Для более удобной работы с лабораторными работами, их можно будет отсортировать по различным критериям. Для этого будут использованы алгоритмы сортировки, работающие с различными классами. Для реализации таких сортировки необходимо реализовать функцию для сравнения объектов одного типа.

Для сортировки будут использованы алгоритмы стандартной библиотеки *C#*.

2.2 Логическое моделирование

Система управления лабораторными работами и отчётами по дисциплине современные языки программирования представляет собой клиент-серверное приложение, в котором общение происходит по *HTTP*.

Сервер представляет из себя приложение *ASP.NET Core* с применением многослойной архитектуры с определенными модулями (слоями):

* *E*ntities, слой который хранит все модели которые используются в приложении;
* DTO, слой с моделями необходимыми для взаимодействия с клиентом;
* Controllers: слой который хранит в себе контроллеры (классы которые предоставляют API для работы);
* Infrastructure: слой который хранит все репозитории и функции необходимые для взаимодействия с базой данных;
* Servises: слой в котором находится все необходимая бизнес логика для работы с данными.

Реализация сервера с помощью ракой архитектуры позволяет облегчить разработку и улучшить расширяемость приложения.

Все модели из слоя *Entities* представлены на диаграмме классов в графической части на листе 2.

В клиенте, который представляет собой *WinForm* приложение есть несколько логический модулей, в архитектуре клиента также есть такие слои как *Entities* и *DTO*, но остальной функционал можно разделить по определенным модулям:

* регистрация и авторизация преподавателя и студента;
* модуль для работы с лабораторными работами;
* модуль для создания отчетов студентами, и их проверки преподавателями.

Реализовав таким образом структуру приложения, можно облегчить разработку и дальнейшее поддерживание, и расширение приложения, как сервера, так и клиента.

2.3 Выбор и обоснование инструментов разработки

В процессе разработки программного обеспечения важно выбрать такие технологии и инструменты, которые обеспечат оптимальную производительность, простоту реализации и дальнейшего сопровождения. Для реализации данного проекта были выбраны следующие инструменты:

1. Язык программирования *C#* был выбран как основной язык программирования по нескольким причинам:

* высокая производительность: C# является языком высокого уровня, который предоставляет мощные средства для разработки производительных приложений. Программы на C# компилируются в промежуточный код (IL), который затем выполняется на платформе .NET, что обеспечивает высокую скорость выполнения;
* обширные возможности: C# предоставляет разнообразные встроенные библиотеки, позволяющие работать с файловыми системами, графическим интерфейсом, сетями и многими другими аспектами разработки. Это упрощает создание приложений разного уровня сложности;
* кроссплатформенность: хотя проект ориентирован на Windows, платформа .NET Core, на которой работает C#, поддерживает кроссплатформенность, что открывает возможности для расширения на другие ОС в будущем;
* сообщество и поддержка: C# – это широко используемый язык программирования с большой базой знаний и поддержкой сообщества, что облегчает поиск решений при возникновении проблем в процессе разработки.

2. *Windows Forms (WinForms): WinForms* был выбран в качестве технологии для создания пользовательского интерфейса (*GUI*) проекта:

* простота и быстрота разработки: WinForms предоставляет средства для быстрой разработки графических интерфейсов с возможностью визуального редактирования форм, что ускоряет процесс создания интерфейсов и уменьшает количество ручного кода;
* широкие возможности кастомизации: хотя WinForms является достаточно простой технологией, он позволяет создавать интерфейсы с высокой степенью настройки элементов, что удобно для создания прикладных программ с уникальным дизайном и функционалом;
* интеграция с C# и .NET: WinForms является частью экосистемы .NET, что обеспечивает удобную интеграцию с языком C# и его библиотеками, упрощая работу с элементами управления, событиями и обработкой данных.

3. *ASP .NET Core*: *ASP.NET Core* был выбран в качестве серверной технологии для создания веб-сервера по следующим причинам:

* заранее заготовленные архитектурные шаблоны, такие как *MVC*, *MVVM*, *Clean* и др.;
* есть значительный опыт в написании веб-серверов на данной технологии с использованием чистой архитектуры
* удобное предоставление *API* во время запуска сервера благодаря технологии *Swagger*, которая поддерживается в *ASP.NET Core*

4. *Entity Framework Core (EF Core)*: *EF Core* был выбран в качестве *ORM (object-relations mapping)* как технология, позволяющая преобразовывать классы *C#* в таблицы реляционной базы данных.

5. Хранение данных в СУБД *PostgreSQL*: *PostgreSQL* был выбран по следующим причинам:

* удобное хранение данных в базе данных (далее БД);
* предоставляет большую надежность информации, по сравнению с файлами;
* есть опыт использования данной БД и приложения *pgAdmin* для взаимодействия с БД;

Использование данных инструментов и технологий позволяет создавать гибкое, производительное и легко поддерживаемое серверное приложение с интуитивно понятным интерфейсом на клиенте и удобной и надежной системой хранения данных.

3 Программная реализация

3.1 Физическая структура

Программное средство «Cистема управления лабораторными работами и отчётами по предмету современные языки программирования» представляет собой клиент-серверное приложение, где сервер и клиент являются независимыми частями программы, а их взаимодействие происходит по унифицированному формату в виде запросов *HTTP*. На сервере и на клиенте описаны *API* и сущности и модели обмена данными (*DTO*), которые передаются в теле запросов.

Каждый запрос выполняется на адресс сервера, и может быть определенного вида:

* *PUT* – запрос подрузумевает отправку данных;
* *POST* – запрос подразумевает отправку данных и получение ответа;
* *GET* – запрос подразумевает получение данных
* *DELETE* – запрос подразумевает удаление данных по запросу.

Пример визуального отображение *API* (набора запросов к серверу) представлен на рисунке 3.1 с помощью инструмента визуализации *Swagger*.

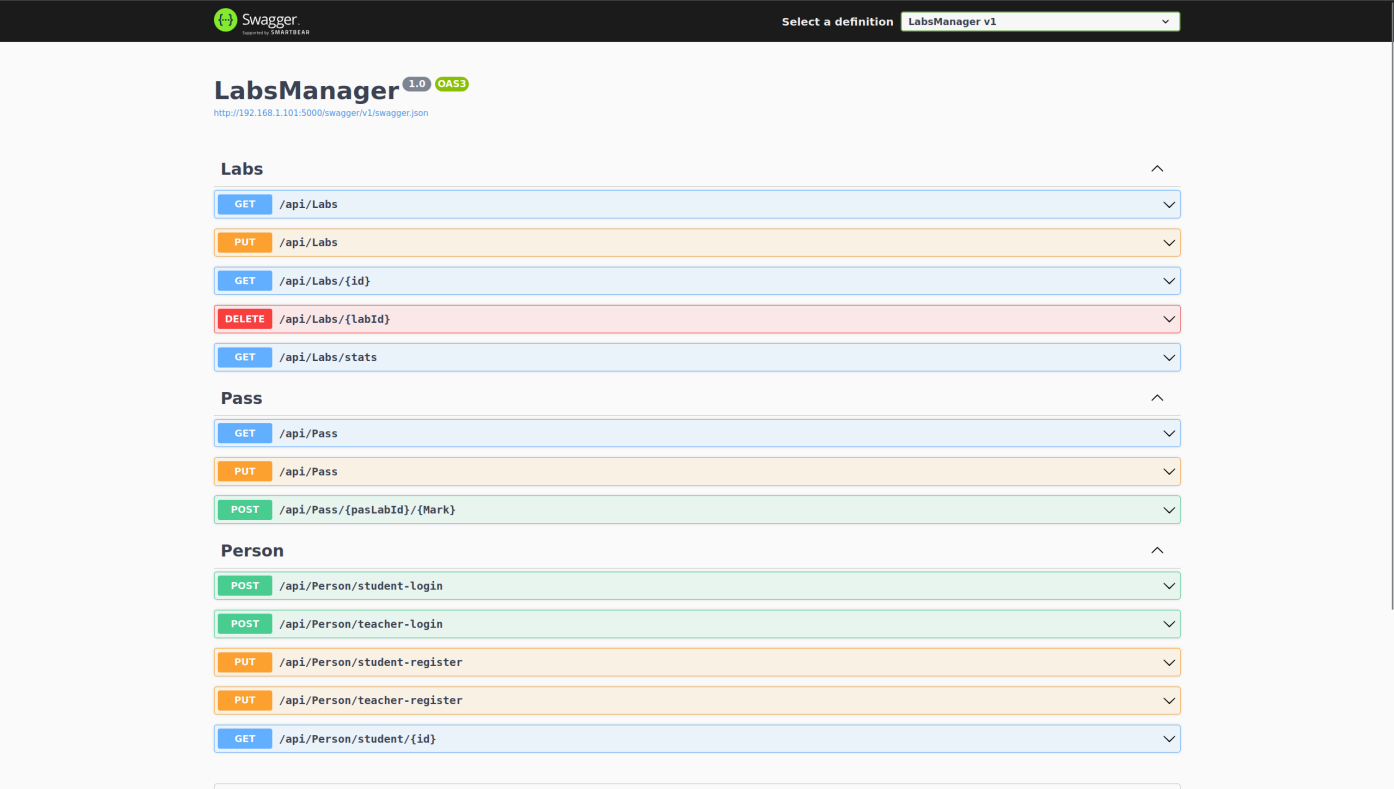


Рисунок 3.1 – *API* сервера в *Swagger*

Сервер представляет собой отдельную программу к которой обращается клиент в виде приложения на *Windows Forms*. Данная структура приложения является современной и позволяет обращаться многим клиентам к одному централихованному мощному серверу.

Клиент же из себя представляет приложение которое в зависимости от выполняемых пользователем действий отправляет разные запросы на сервер, клиент не хранит никаких данных, что позволяет достичь независимости от устройства на котором запускают приложение.

3.2 Описание разработанных модулей

Программное средство «Cистема управления лабораторными работами и отчётами по предмету современные языки программирования», как было описано выше, состоит из клиент и сервера.

Сервер предстовляет собой *C#* приложение написанное с использованием чистой многослойной архитектуры,то есть имеет следующие модули:

* *Controllers*: слой, который хранит в себе контроллеры, содержащие *API*, необходимые *DTO* модели, модели которые придут в ответ и *URL* запросов;
* *DTO*: слой, который хранит в себе сущности для обмена, сущности необходимые для приема запросов и сущности для ответов сервера;
* *Entities*: слой, который хранит основные сущности приложения с которыми взаимодействует пользователь;
* *Infrastructure*: слой, который хранит в себе методы обращения к БД – репозитории и файл для создания миграций;
* *Migrations*: слой, который хранит миграции приложения, которые необходимы для инициализации БД;
* *Services*: слой, который хранит основную логику приложения, здесь описаны все проверки, манипуляции и преобразования.

Также поверх всех слоем идут такие файлы как *Program.cs, appsettings.json, launchSettings.json* необходимые для инициализации и первоначальной автоматической настройки сервера.

Файловая структура сервера представлена на риисунке 3.2.

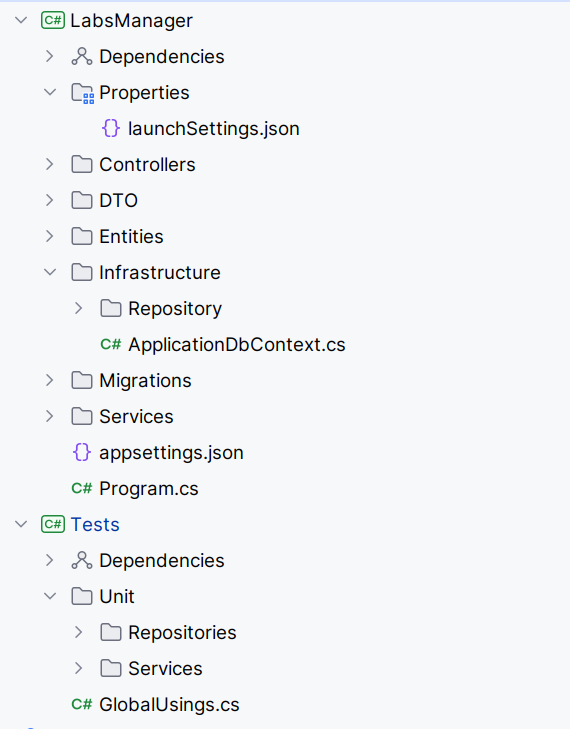


Рисунок 3.2 – Файловая структура сервера

Схема базы данных представлена на рисунке 3.3.

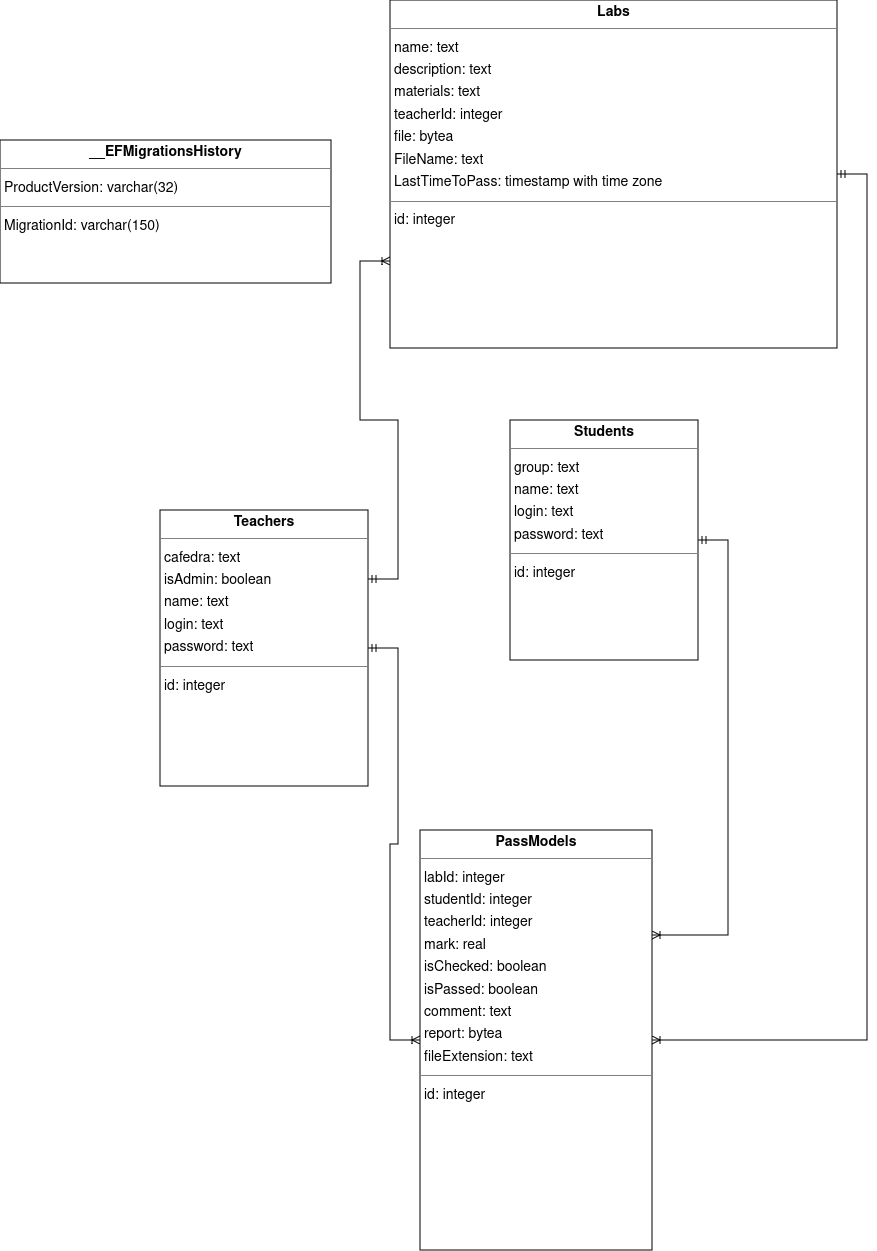


Рисунок 3.3 – Схема базы данных

Применение принципов чистой архитектуры улучшает читаемость кода, упрощает расширение кодовой базы приложения и позволяет логически разделить модули программы по слоям.

Клиент представляет собой *Windows Forms* приложение с использованием нескольких структурно файловых особеннойтей:

* Папка *Domain,* которая хранит в себе элементы чистой архитектуры *Entity* и *DTO;*
* Папка *Data,* где хранятся все выводимые приложениием данные (статистика);
* Наличие файла *ENV.cs*  или файла общих переменных, в котором хранятся такие важные вещи как базовый *URL* сервера и базовая папка для вывода статистики.

Остальное пространстро занимаю сами формы с их логикой.

Такой подход к написанию клиента позваляет легко и эффективно делать запросы к единому серверу. При этом сами формы отвечяют за определенные действия в приложении, их и вся файловая структура представлены на рисунке 3.3.

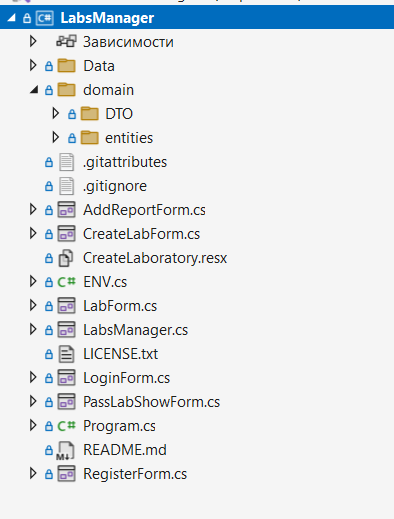


Рисунок 3.3 – Файловая система клиента

Каждая форма отвечает за определенные логические действия в приложении, что можно соотносить с модулями программы.

Перечень логических модулей:

* работа с лабораторными;
* работа с проверками;
* работа с пользователями.

Логические модули отличаются от фактических форм, но в одном модуле может быть разное количество форм. Такой подход позволяет легко расширять уже существующие модули посредством добавления новых форм.

4 Тестирование

При создании программных продуктов необходимо поддерживать качество написанного кода и правильность изменяемых функций. Для этих целей разработчики пишут модульные тесты, которые позволяют отслеживать правильность выполнения функци после каждого изменения кода.

С этой целью были разработаны модульные тесты только для серверной части приложения, так как клиентское приложение лишь отправляет запросы на сервер.

В качестве технологии тестирования была выбрана библиотека *xUnit,* которая была специально созданна для написания тестов в *C#* приложениях. Эта технология позволяет лекго и быстро создавать тестовые классы которые тестируют определенную функцию. Также вместе с *xUnit* были использованы технологии *FluentAssertion* и *Moq*. *FluentAssertion* позволяет писать более читаемые проверки для подтверждения возращаемого значения функцией а *Moq*  позволяет использовать заглушки зависимостей, для обеспечения работы только целевой функции.

Для обеспечения правильной работы с базой данных были написаны тесты для репозиториев и для правильной работы бизнес-логики приложения были написаны тесты для сервисов. На рисунке 4.1 представлен пример тестового класса для репозитория, где в конструкторе инициализируется база данных в оперативной памяти, для обесмечения быстрого доступа, а в методах проверяется правильность выполнения операций к базе данных. на рисунке 4.2 представлен тестовый класс для тестирования методов сервиса. Где вначале создаются заглушки зависимостей, а далее в конструкторе они инициализируются.



Рисунок 4.1 – Тестовый класс репозитория



Рисунок 4.2 – Тестовый класс сервиса

Пример выполненых тестов показан на рисунке 4.3.

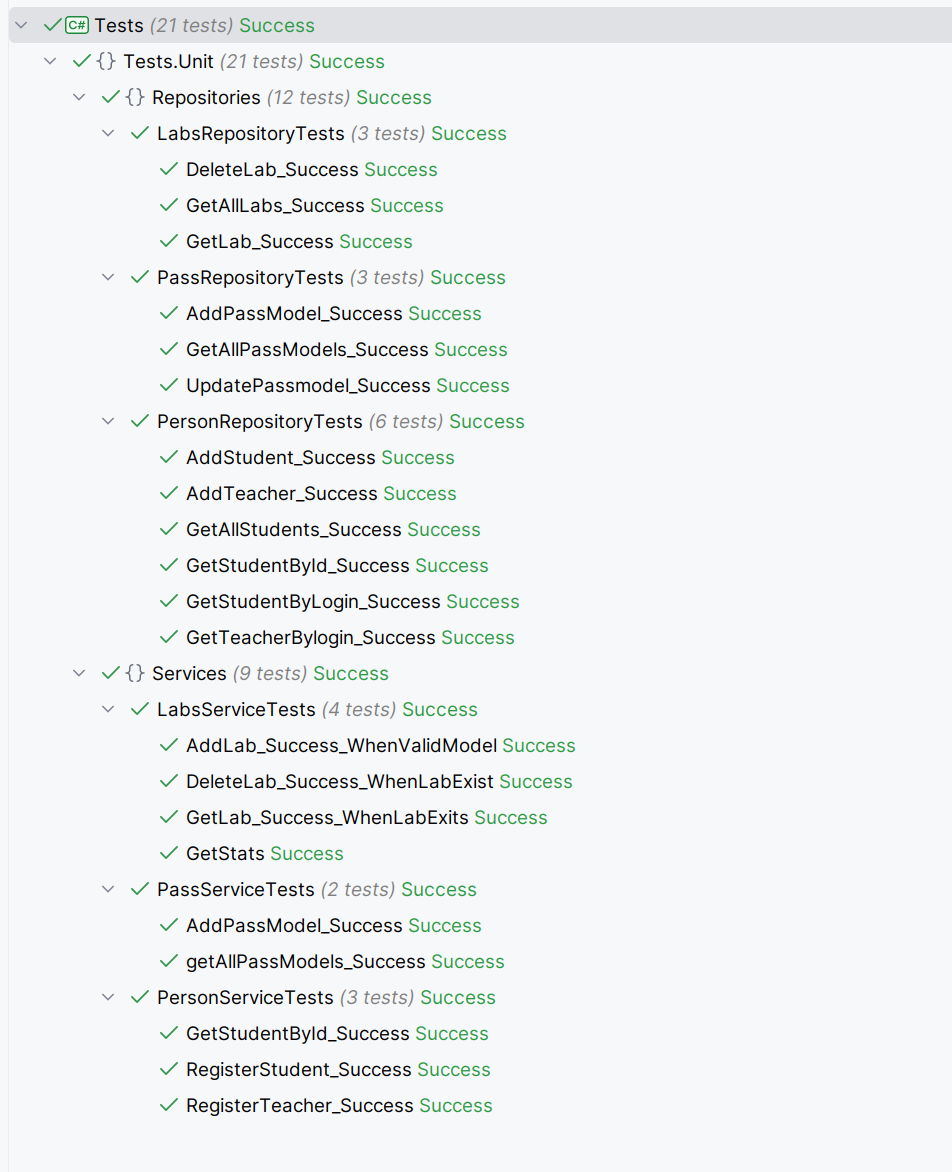


Рисунок 4.3 – Результат выполнения тетстов

Такой подход позволяет легко автоматизировать отслеживание качества выполненных функция путем запуска модульных тестов при каждом изменении кода иои перед запуском приложения прямо внутри *Visual Studio*.

5 Применение программы

5.1 Руководство пользователя

Программоное средство «Система управления лабораторными работами и отчётами по предмету современные языки программирования» используется слудующим образом:

* авторизация;
* работа с лабораторными.

Приложение запускается с выводом окна авторизации (Рисунок 5.1), куда необходимо ввести в соответствующие поля свой логин и пароль для авторизации.

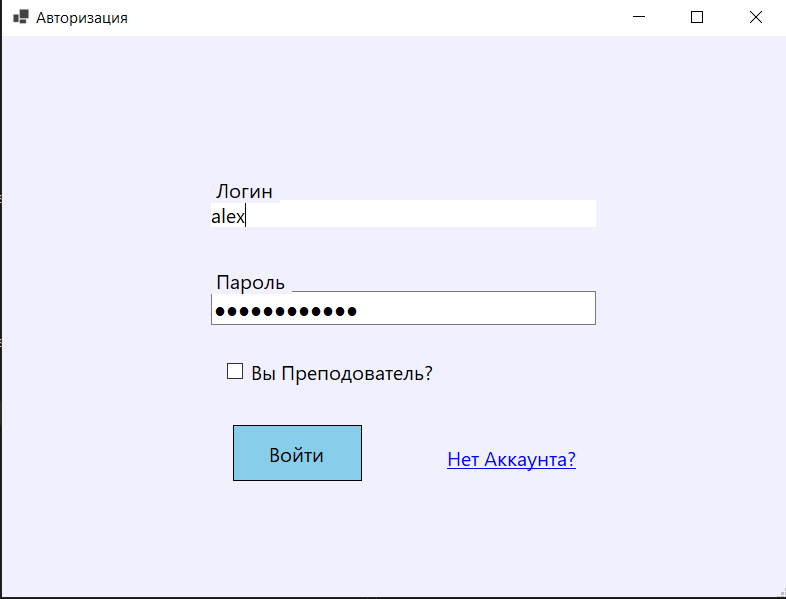


Рисунок 5.1 – Окно авторизации

Если у пользователя еще нет зарегистрированного аккаунта, то ему необхадимо нажать на текст «Нет Аккаунта» правее кнопки авторизации, для открытия окра регистрации (Рисунок 5.2), в котором пользователь заполняет все поля и нажимает на кнопку «Зарегистрироваться».

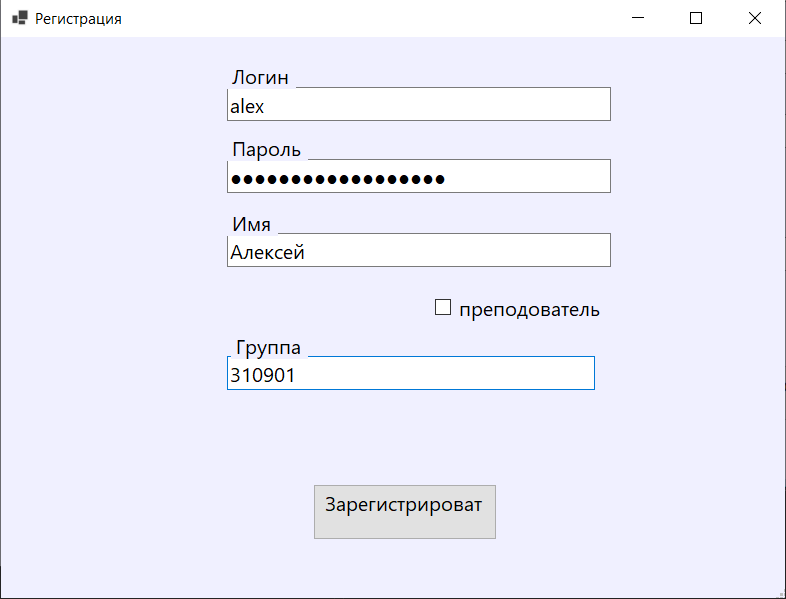


Рисунок 5.2 – Окно регистрации

После прохождения регисрации пользователя необходимо будет в окне авторизации ввести свой логин и пароль, после чего откроется главный экран приложения (Рисунок 5.3), где можно выбрать необхадимую для работы вкладку: профиль (открывается по умолчанию), все лабы (список всех лабораторных работ), на проверке(список лабораторных работ на которые студенты сделали отчет и они не проверены), проверены (проверенные отчеты по лабораторным работам). В соответствии с выбором откроется соответствующая вкладка на главном экране.

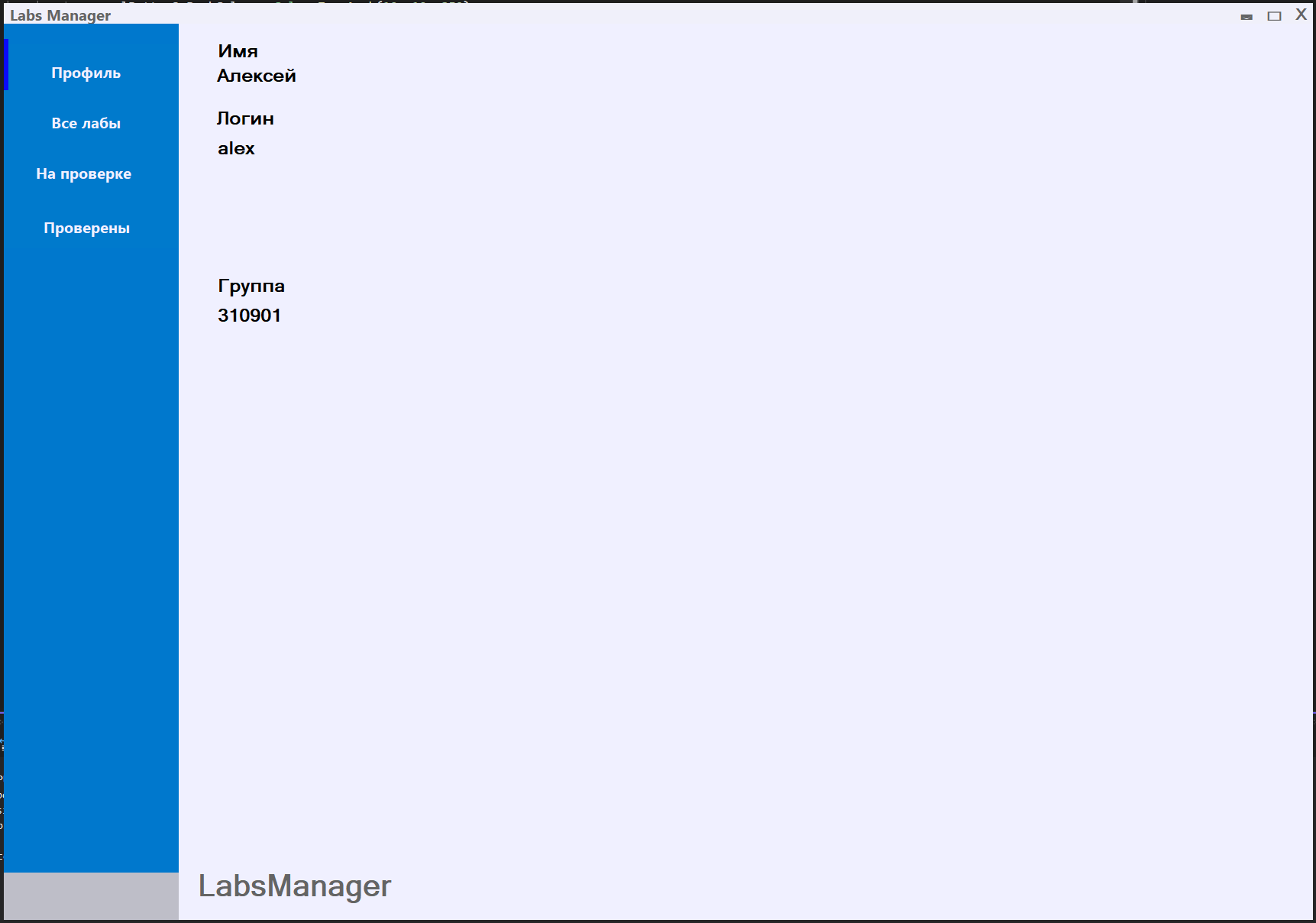


Рисунок 5.3 – Главный экран приложения

Для создания отчета по лабораторной работе необходимо сначала перейти на вкладку все лабы, выбрать необходимую лабораторную работу, после чего откроется окно лабораторной работы (Рисунок 5.4), где можно ознакомится с описанием и материалами к лабораторной работе и скачать задание оставленное преподователем.

Далее после создания отчета на своем компьютере необходимо нажать на кнопку добавить отчет, после чего откроется окно для загрузки отчета к лабораторной работе (Рисунок 5.5). Далее необходимо нажать на кнопку Загрузить и в всплывающем окне выбрать свой отчет, после чего справа должен появиться путь к файлу. После чего необходимо нажать на кнопку отправить,после чего отчет добавится к лабораторной работе и его сможет проверить преподователь.

Для просмотра своих отправленных отчетов можно перейти в вкладки на проверке и проверено где при нажатии на свой отчет откроется его форма обзора (Рисунок 5.6).

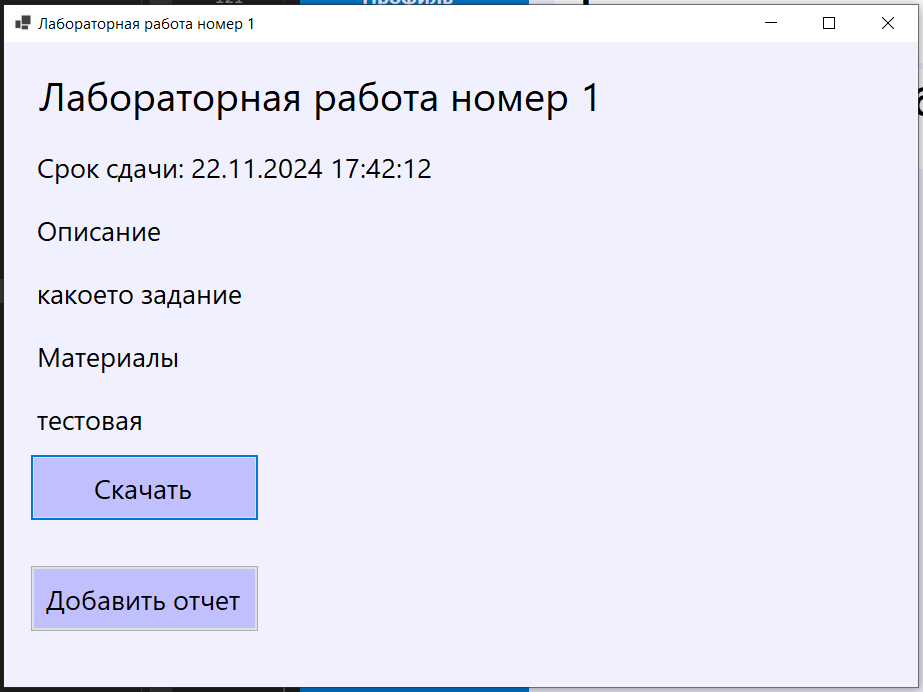


Рисунок 5.4 – Окно лабораторной работы

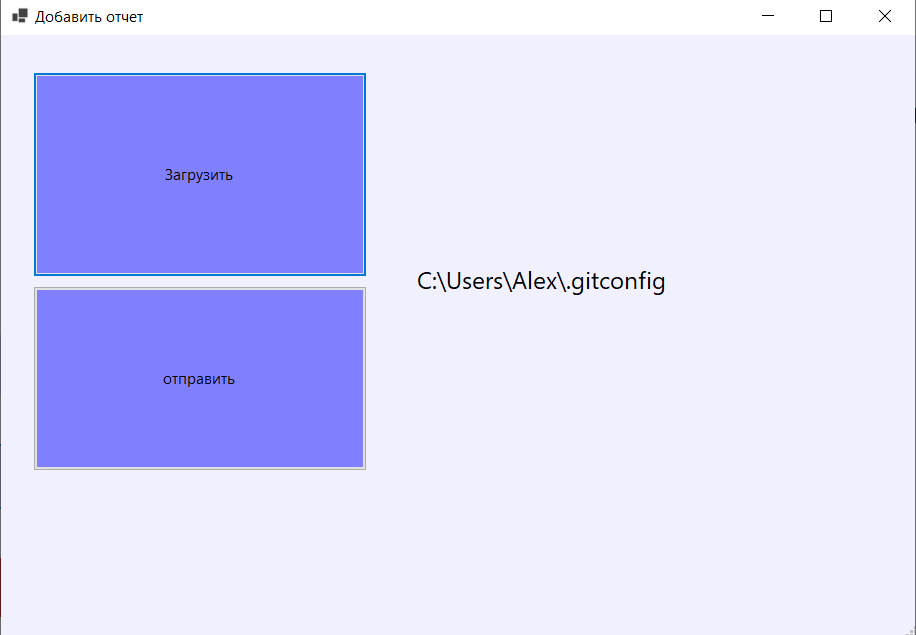


Рисунок 5.5 – Окно загрузки отчета

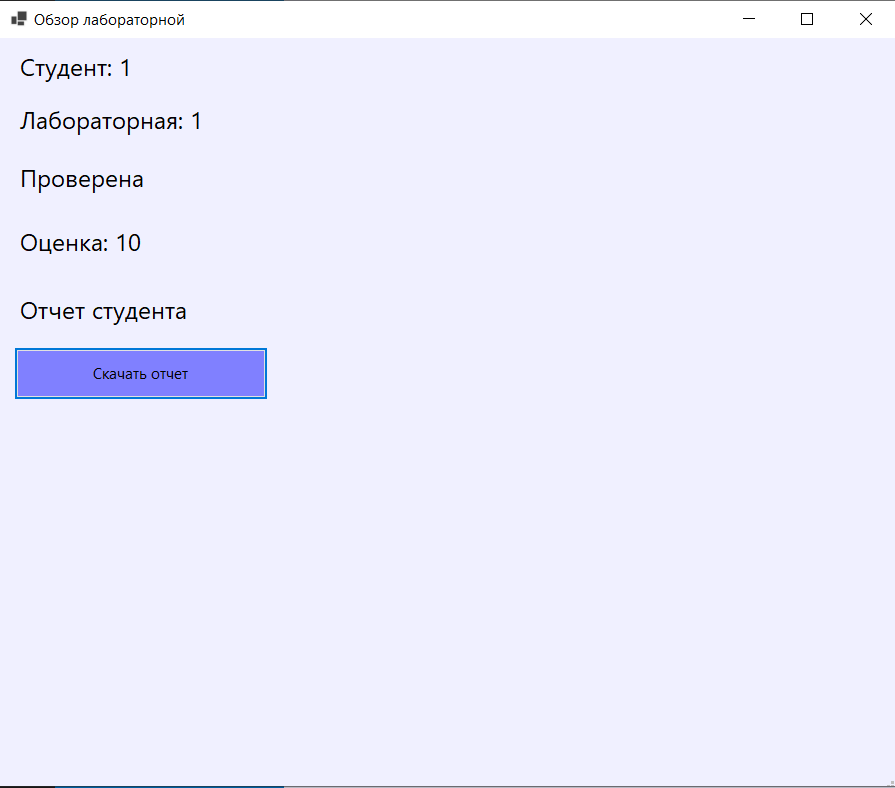


Рисунок 5.6 – Окно обзора отчета

Если в систему входит преподователь, то он может создать лабораторную работу, которая будет отображаться во вкладке все лабы. Для этого необходимо перейти во вкладку все лабы и справа снизу нажать на кнопку создать лабу, после чего откроется окно создания лабораторной работы (Рисунок 5.7).

Необходима заполинть все поля, загрузить задание, нажав на кнопку Загрузить, после чего появится путь к файлу, и выбрать срок сдачи лабораторной работы. После этого для создания лабораторной работы нужно нажать на кнопку создать, после чего лабораторная работа будет создана

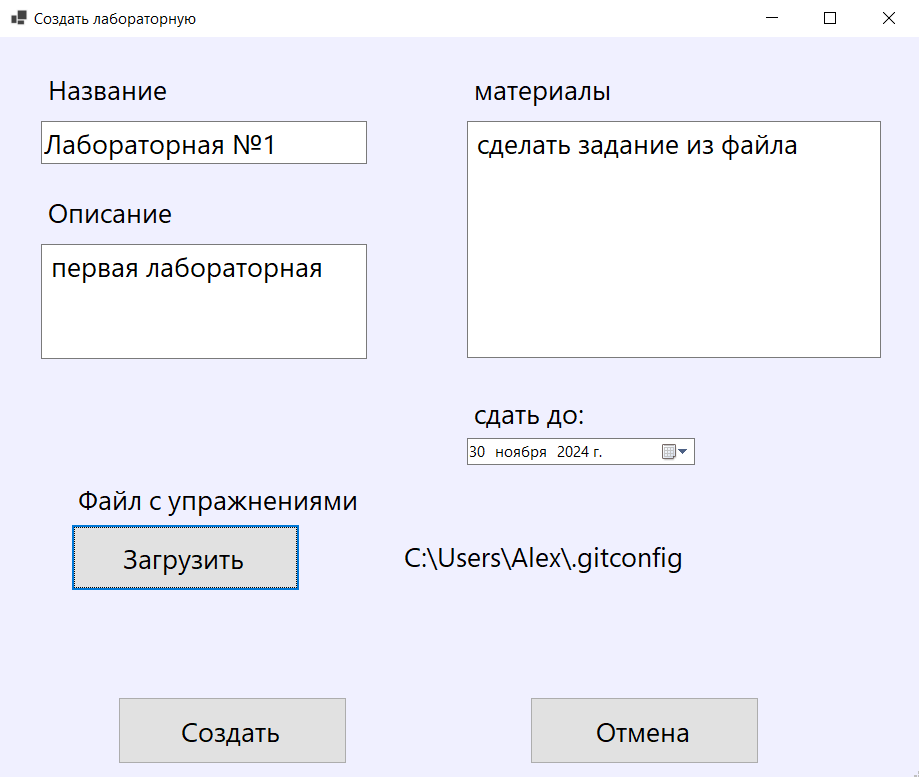


Рисунок 5.7 – Окно создания лабораторной работы

Для проверки лабораторной работы преподователем необходимо в главном окне перейти во вкладку на проверке и выбрать необходимую лабораторную работу, после чего откроется окно обхора лабораторной работы для преподователя (Рисунок 5.8), где можно скачать отчет сдудента и выставить оценку за выполнение лаабораторной работы, по умолчанию лабораторная работа считается сданой, если оценка 4 и выше.

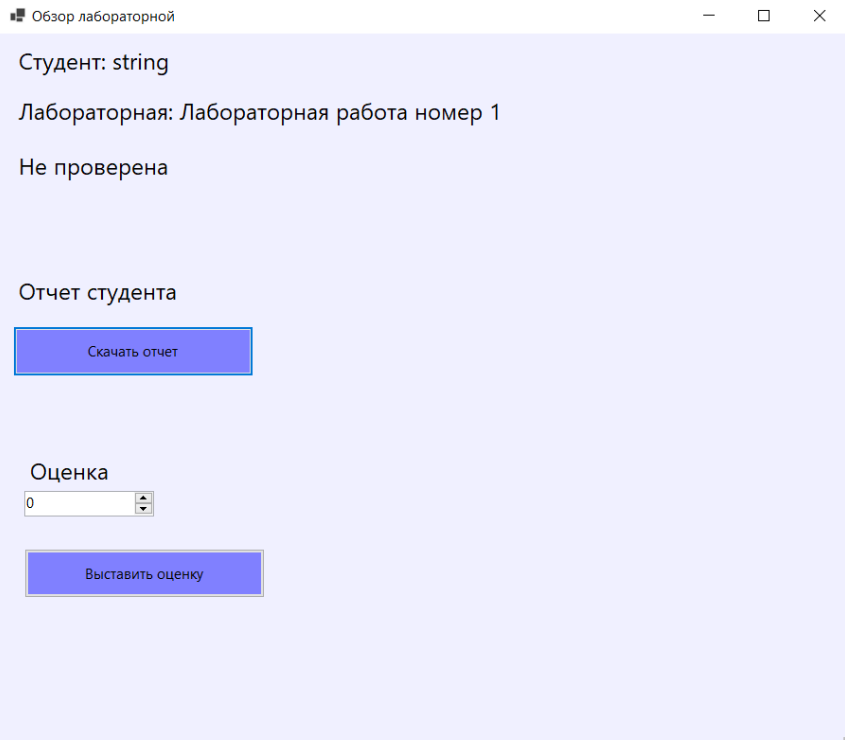


Рисунок 5.8 – Окно обзора лабораторной работы

При нажатии на кнопку скачать отчет, он будет загружен по выбранному пути. Если нажать на кнопку выставить оценку, то система выставвит оценку студенту и обновит статус лабораторной работы.

Список использованных источников

1. Стандарт предприятия [Электронный ресурс]. – Режим доступа: СТП\_2024.pdf.
2. LeetCode [Электронный ресурс]. – Режим доступа: https://leetcode.com/.
3. Инструмент для создания графических материалов [Электронный ресурс]. – Режим доступа: https://app.diagrams.net/.
4. Обучение *C#*, *UTF-8* с первой программы [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/731614/.
5. *Moodle* [Электронный ресурс]. – Режим доступа: https://moodle.org/.
6. Руководство по использованию приложения консоли *cmd* [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cmd/.
7. Справочник по языку *C#* [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/csharp/csharp/csharp-language-reference/.
8. Учебник по программному обеспечению *Visual Studio* [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/visualstudio/windows/.
9. Знакомство с исключительными ситуациями [Электронный ресурс]. – Режим доступа: http://www.cyberguru.ru/sources/cpp/classes-mfc/isklyuchitelnye-situatsii.html.
10. *Google Classroom* [Электронный ресурс]. – Режим доступа: https://classroom.google.com/.
11. Варианты на все случаи жизни: как написать полезный use case [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/chto-takoe-use-case-kak-ih-napisat/.
12. *UML*: обзор основных типов диаграмм. Часть 2 [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/756552/.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода