  
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ  
Департамент программной инженерии и искусственного интеллекта**

**РАЗРАБОТКА СИСТЕМЫ «To-Do List»**

КУРСОВОЙ ПРОЕКТ  
по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению 09.03.04 «Программная инженерия»

Выполнили:  
студенты гр. Б9120-09.03.04прогин  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Перепечин В.В.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Алексеев В.М.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Нельбасов Д.М. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Колесник А.С.   
Руководитель:  
Ассистент департамента ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю.С.

г. Владивосток  
2024

[Введение 5](#_Toc156212253)

[1. План проекта 6](#_Toc156212254)

[2. Регламент проведения инспекции 7](#_Toc156212255)

[2.1) Критерии отнесения к формальной / неформальной инспекции для различных типов рабочих продуктов. 7](#_Toc156212256)

[2.2) Перечень ролей участников инспекции и их обязанности, число участников в зависимости от объёма рабочего продукта 8](#_Toc156212257)

[2.3) Этапы инспекции 8](#_Toc156212258)

[2.4) Планировка инспекции 8](#_Toc156212259)

[2.5) Порядок подготовки к инспекции 9](#_Toc156212260)

[2.6) Порядок проведения инспекции 10](#_Toc156212261)

[2.7) Порядок верификации учета замечаний 12](#_Toc156212262)

[2.8) Метрики, характеризующие эффективность инспекций 13](#_Toc156212263)

[2.9) Перечень статусов и степени важности замечаний 14](#_Toc156212264)

[3. Модель состояний задач 16](#_Toc156212265)

[• New (Новая) 16](#_Toc156212266)

[• Planned (Запланировано) 16](#_Toc156212267)

[• In Progress (В процессе) 16](#_Toc156212268)

[• Paused (На паузе) 16](#_Toc156212269)

[• Review (На проверке) 16](#_Toc156212270)

[• Completed (Завершено) 16](#_Toc156212271)

[• Closed (Закрыто) 17](#_Toc156212272)

[Правила создания новой задачи 17](#_Toc156212273)

[Правила перехода задачи из состояния в состояние 17](#_Toc156212274)

[4. Требования к проекту 18](#_Toc156212275)

[Требования к подсистеме «Пользовательский интерфейс» 18](#_Toc156212276)

[Требования к подсистеме «Подсистема аутентификации и регистрации» 19](#_Toc156212277)

[Требования к подсистеме «Подсистема создания задач» 19](#_Toc156212278)

[Требования к подсистеме «Подсистема Ввода-Вывода» 19](#_Toc156212279)

[Требования к подсистеме «Подсистема Редактирования и Удаления Задач» 19](#_Toc156212280)

[Требования к подсистеме «Подсистема Отображения Задач» 20](#_Toc156212281)

[Требования к подсистеме «Подсистема Статистики» 20](#_Toc156212282)

[Требования к подсистеме «Подсистема Уведомлений и Напоминаний» 20](#_Toc156212283)

[Требования к подсистеме «Подсистема импорта/экспорта данных» 20](#_Toc156212284)

[5. Архитектура проекта 22](#_Toc156212285)

[Архитектурно-контекстная диаграмма 22](#_Toc156212286)

[Диаграмма классов 22](#_Toc156212287)

[Классы контроллеры 23](#_Toc156212288)

[Классы представления 25](#_Toc156212289)

[Классы модели 25](#_Toc156212290)

[6. Измерения проекта 27](#_Toc156212291)

[Метрика эффективности процесса производства 27](#_Toc156212292)

[IFD (Integrated Function Deployment) 27](#_Toc156212293)

[FS (False Scrining) 27](#_Toc156212294)

[IPF (Integrated Product Functionality) 27](#_Toc156212295)

[7. Рекомендации по кодированию 29](#_Toc156212296)

[8. План тестирования проекта 31](#_Toc156212297)

[9. Интерфейс прототипа 34](#_Toc156212298)

[Вид для не авторизированного пользователя 34](#_Toc156212299)

[Вид для авторизированного пользователя 35](#_Toc156212300)

[Вид для привилегированного пользователя 38](#_Toc156212301)

[Заключение 41](#_Toc156212302)

[Список литературы 42](#_Toc156212303)

# Введение

Развитие индустрии информационных систем акцентируется на различных аспектах, включая проектирование, разработку, и тестирование. Этот процесс требует взаимодействия специалистов различных областей и, для обеспечения эффективной коммуникации и четкого распределения обязанностей, важно применять стандартизированные методы.

Наша тема "To-Do List" фокусируется на создании программного средства для управления задачами. В рамках данного проекта мы сталкиваемся с задачей коллективной разработки, где основной целью является не только создание самого инструмента, но и разработка соответствующей технической документации. Таким образом, главной задачей проекта "To-Do List" является успешная реализация программного средства, применяя принципы коллективной промышленной разработки. Это позволит обеспечить эффективную координацию работы команды и гарантировать соответствие конечного продукта заявленным требованиям.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать план проекта.
2. Разработать регламент проведения инспекции.
3. Разработать модель состояний задач.
4. Разработать требования к проекту.
5. Разработать архитектуру проекта.
6. Разработать измерения проекта.
7. Разработать рекомендации по кодированию.
8. Разработать план тестирования проекта.
9. Разработать интерфейс проекта

# План проекта

План проекта – это документ, содержащий подробную информацию о проекте: исполнителях, задачах и сроках. Документ является конечным результатом этапа планирования, утверждается до начала любых работ и становится самым главным и достоверным источником информации о грядущем проекте.

В нашем случае исполнителями являются следующие лица:

* Руководитель – Перепечин В.В.;
* Кодер – Колесник А.С.;
* Программный архитектор – Алексеев В.М.;
* Технический писатель – Нельбасов Д.М..

На рисунке 1 представлен перечень задач для выполнения и примерные сроки их реализации.

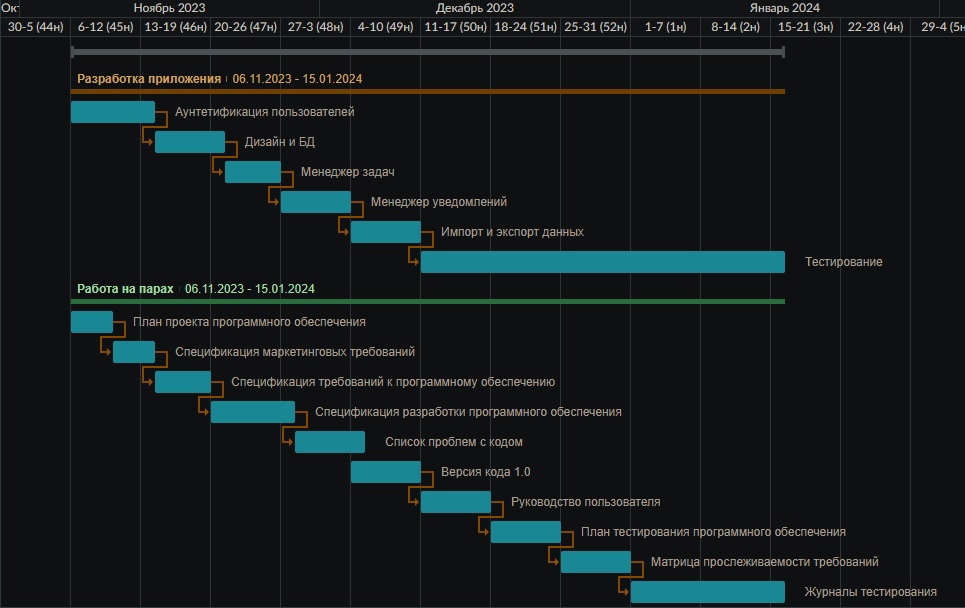


Рисунок 1 – План проекта

# Регламент проведения инспекции

Верификация рабочих продуктов является неотъемлемой частью процесса обеспечения их качества. Современные технологии программирования предлагают специальные стандарты, подходы и механизмы для проведения верификации рабочих продуктов, включая инспекции.

* Инспекция — это мероприятие, направленное на обеспечение качества рабочих продуктов проектов по разработке программного обеспечения и других деятельностей, которое проводится разработчиками с возможным участием представителей заказчика.
* Концептуально инспекция имеет следующие цели:
* Выявление ошибок в функциях, логике, содержании или реализации рабочих продуктов на ранних этапах разработки и предотвращение их наследования;
* Эффективное донесение концепции или реализации продукта до всех заинтересованных сторон (через их участие);
* Оптимизация, оценка или улучшение рабочего продукта.

## 2.1) Критерии отнесения к формальной / неформальной инспекции для различных типов рабочих продуктов.

|  |  |
| --- | --- |
| **Рабочий продукт** | **Размер продукта или изменения в**  **нем, не менее** |
| Требования | 5 страниц или 10% рабочего продукта |
| Документы по дизайну | 5 страниц или 10% рабочего продукта |
| Не комментированный код | 1000 NCLOC |
| Тесты (Test Cases) | 5 страниц или 10% рабочего продукта |

## 2.2) Перечень ролей участников инспекции и их обязанности, число участников в зависимости от объёма рабочего продукта

Каждый участник инспекции играет определенную роль.

Роли участников формальной инспекции это:

Автор (Перепечин В.В) - сотрудник, разработавший инспектируемый рабочий продукт, либо сделавший инспектируемые изменения в существующем рабочем продукте;

Председатель, инспектор (Нельбасов Д.М.) - сотрудник, выполняющий роль председателя инспекции и ответственный за эффективную проверку инспектируемого рабочего продукта;

Секретарь (Колесник А.С.) - сотрудник, ответственный за создание и распространение документации по инспекции;

Ведущий (Алексеев В.М.) - сотрудник, представляющий рабочий продукт инспекторам;

## 2.3) Этапы инспекции

1) Планирование инспекции.

2) Назначение инспекции.

3) Обзорное собрание

4) Подготовка к инспекции.

5) Собрания по инспекции.

Завершение инспекции (распространение результатов, переработка рабочего продукта, проверка исправления недостатков в рабочем продукте).

## 2.4) Планировка инспекции

При планировании инспекции коллективно выбирается дата, время, формат (очный или заочный) и платформа (при заочной инспекции) проведения инспекции.

## 2.5) Порядок подготовки к инспекции

5.1) Функции автора

• Инициация инспекции: Инициировать формальную инспекцию программы "To-Do List". Для этого, как автор, Перепечин В.В должен уведомить руководителя проекта о готовности программы к инспекции и зафиксировать стабильную версию продукта. С этого момента и до собрания не должны вноситься изменения в программу.

• Обзорное собрание: При необходимости провести обзорное собрание для введения участников инспекции в курс дела, объяснить суть и особенности программы "To-Do List".

5.2) Функции председателя

• Проверка готовности: Убедиться, что программа "To-Do List" готова к инспекции, используя чек-лист, соответствующий типу рабочего продукта.

• Обзорное собрание: Определить необходимость проведения обзорного собрания и сказать об этом автору.

• Приглашения: Проверить, что все участники инспекции подтвердили участие или предоставили обоснованный отказ. В случае получения отказов, председатель должен оценить возможность проведения инспекции без данного участника.

• Анализ протоколов и принятие решений:

• Провести инспекцию: Если инспектор готов и обнаруженные проблемы не слишком велики или в основном являются комментариями.

• Перенести инспекцию: Если инспектор не готов или время подготовки не соответствует сложности продукта.

• Отменить инспекцию: Если найденное количество проблем велико, и они серьезные или спорные, касающиеся содержания или функциональности программы.

• Уведомление участников: Председатель должен незамедлительно уведомить всех участников о принятом решении.

5.3) Общие обязанности председателя и автора

• Организация собрания инспекции: Определить место, время и продолжительность собрания инспекции. Длительность не должна превышать 2 часа. Если необходимо, распределить инспекцию на несколько собраний.

• Уведомление участников: Оповестить участников о размере инспектируемого продукта и рекомендуемом времени подготовки.

5.4) Функции секретаря

• Документация: Создать и распространить документацию по инспекции, включая детали собраний и результаты проверок.

5.5) Функции ведущего

• Участие в инспекции: Принять активное участие в инспекционных собраниях, предоставляя конструктивную обратную связь.

5.6) Функции инспектора

• Ответ на приглашение: Принять или отклонить приглашение на инспекцию, указав причины отказа при необходимости.

• Ограничение участия: Ограничить количество инспекций до двух в день для эффективности работы.

## 2.6) Порядок проведения инспекции

6.1) Функции председателя

• Оглашение цели и представление участников: Огласить цель собрания, представить участников инспекции и уточнить их роли.

• Проверка замечаний инспектора: Убедиться, что у каждого инспектора есть замечания и попросить ведущего начать представление программы "To-Do List".

• Контроль за ходом собрания: Следить за темпом собрания, пресекать попытки решения проблем на месте, разрешать конфликты, и следить за сосредоточением на выявлении недостатков программы.

• Принятие решений по замечаниям: В случае разногласий при инспекции, принимать решения относительно статуса замечаний.

• Прерывание или перенос собрания: Прервать собрание, если оно превышает запланированное время, или перенести при нарушении участниками порядка.

• Назначение проверяющего: В конце собрания принять решение о назначении проверяющего для контроля изменений в программе.

6.2) Функции ведущего

• Представление программы: Эффективно представить программу "To-Do List" участникам собрания, выбрав наиболее подходящий метод представления.

6.3) Функции секретаря

• Формулировка замечаний: Огласить окончательную формулировку каждого замечания, его статус и местоположение.

• Обработка замечаний: Секретарь анализирует замечания в протоколах, выявляя и объединяя повторяющиеся проблемы рабочего продукта, чтобы составить полный список всех обнаруженных проблем.

6.4) Функции инспектора

• Изучение продукта: инспектор должен самостоятельно изучить программу "To-Do List", применяя свой опыт, стандарты, руководства и контрольные списки.

• Заполнение протокола: Инспектор должны заполнить протокол инспекции, указывая время подготовки и описание обнаруженных ошибок, и отправить его председателю, секретарю и автору.

• Оглашение замечаний: Задавать вопросы и оглашать найденные проблемы в программе.

• Высказывание мнений: Высказывать мнение о статусе каждого замечания, корректируя их при необходимости во время собрания.

• Мнение о реинспекции: В конце собрания высказать мнение о необходимости повторной инспекции программы.

6.5) Функции Автора

• Анализ протоколов: Автор должен ознакомиться с замечаниями, изложенными в протоколах инспектора, и проанализировать их до собрания.

• Ответы на вопросы: Давать четкие ответы на вопросы инспектора, без попыток оценить или изменять рабочий продукт во время собрания.

6.6) Общее решение

• Решение о реинспекции: На собрании коллегиально принимается решение о необходимости повторной инспекции программы "To-Do List", если это необходимо.

## 2.**7) Порядок верификации учета замечаний**

7.1) Функции автора

• Анализ и исправление: Проанализировать все замечания, зафиксированные в протоколе собрания, и исправить все недостатки в программе "To-Do List".

• Фиксация решений: Решение по каждому замечанию должно быть зафиксировано в протоколе инспекции.

7.2) Функции секретаря

• Рассылка протокола: Разослать всем участникам формальной инспекции протокол собрания. Секретарь должен уведомить всех участников и заинтересованных лиц о готовности протокола, приложив его или указав место его нахождения. Если в ходе подготовки и собрания не было выявлено замечаний, это также должно быть отражено в протоколе.

7.3) Функции председателя

• Проверка протокола: Проверить протокол инспекции на правильность и полноту заполнения, включая формулировки и статусы замечаний. Убедиться, что протокол был разослан всем участникам инспекции и заинтересованным лицам.

7.4) Функции инспектора

• Финальная проверка: Проверить, что все замечания должным образом учтены в обновленной версии программы "To-Do List". Формальная инспекция считается завершенной только после утверждения проверяющим.

## 2.8) Метрики, характеризующие эффективность инспекций

Данные обо всех формальных инспекциях, проводимых на предприятии, должны документироваться и использоваться для оценки эффективности формальных инспекций при разработке программных продуктов. Эти данные используются также для оценки и усовершенствования процесса организации и проведения формальных инспекций.

Набор метрик по инспекциям, собираемых на конкретном проекте, определяется нормативными документами по процессу и программой измерений предприятия и планируется на фазе планирования проекта.

Inspection Fault Density (IFD)

IFD = (Количество найденных ошибок / Размер рабочего продукта) Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – инспекция, измеряемый атрибут – плотность найденных в ходе инспекции ошибок.

Единица измерения – ошибка / <страница, требование, LOC, тест>.

Процесс "inspection" в этом контексте означает формальную, часто ручную проверку исходного кода на предмет ошибок или других проблем. Это может включать в себя код-ревью, где другие разработчики проверяют код на соответствие стандартам программирования, а также на наличие логических и синтаксических ошибок.

Вот несколько ключевых аспектов Inspection Fault Density:

• Мера качества:

Высокая плотность дефектов может указывать на проблемы в процессах разработки или на недостаточное тестирование. Напротив, низкая плотность дефектов обычно свидетельствует о хорошем качестве кода.

• Планирование и улучшение процессов:

Эта метрика может использоваться для планирования будущих проектов и улучшения процессов разработки. Она помогает определить, какие области кода требуют больше внимания и ресурсов на тестирование и исправление.

• Сравнительный анализ:

Inspection Fault Density также может использоваться для сравнения эффективности разных команд разработчиков или различных проектов внутри организации.

• Улучшение:

Регулярное использование этой метрики позволяет командам проактивно идентифицировать и исправлять проблемы в коде до того, как они станут серьезными.

Так, метрика IFD характеризует эффективность инспекции, а также качество инспектируемого продукта. Чем больше IFD, тем эффективнее инспекция при неизменном качестве рабочего продукта. Чем меньше IFD, тем выше качество продукта при условии неизменной эффективности инспекции. Целью предприятия является снижение IFD, чем меньше IFD, тем лучше.

Метрика используется для принятия решения об эффективности инспекции и принятия решения о необходимости реинспекции.

## 2.9) Перечень статусов и степени важности замечаний

• Дефект - проблема, которая найдена на фазе, отличной от той, на которой внесена.

• Ошибка – проблема, которая найдена на той же фазе, на которой внесена.

• Комментарий – это наблюдение, предложение, рекомендация или улучшение, предложенное для будущего выпуска рабочего продукта или вопрос, требующий разъяснения. Внесение изменения в рабочий продукт в соответствии с комментарием – это результат договоренности автора рабочего продукта и автора комментария.

• Замечание для исследования – проблема, природа которой не может быть определена на собрании и требует дополнительного исследования. В результате дополнительного исследования такая проблема должна получить одно из вышеупомянутых значений статусов

Допустимые значения степени серьёзности замечания:

• Критическая (critical)

• Особо важная (major)

• Средняя (moderate)

• Мелкая, незначительная (minor)

• Другие (other)

# Модель состояний задач

Каждая задача, являясь отражением делового процесса, проходит определенные состояния. Сначала идет создание задачи, потом идет выполнение работ по задаче, после выполнения задача завершается.

### • New (Новая)

Описание: Задача создана, но работа по ней еще не начата.

Применение: Когда пользователь создает новую задачу в системе, она автоматически получает статус "New".

### • Planned (Запланировано)

Описание: Задача запланирована, но еще не начата. В это время задача может быть распределена по приоритетам, назначена на конкретные даты.

Применение: Пользователь или система управления задачами назначает задаче дату выполнения и приоритет.

### • In Progress (В процессе)

Описание: Работа над задачей активно ведется.

Применение: Пользователь начинает выполнение задачи и переводит её в состояние "In Progress".

### • Paused (На паузе)

Описание: Временная приостановка работы над задачей.

Применение: Если задача не может быть завершена сразу или требует дополнительной информации/ресурсов, она может быть переведена в состояние "Paused".

### • Review (На проверке)

Описание: Задача ожидает проверки или одобрения, если это требуется по условиям работы.

Применение: После завершения задачи, если требуется, она отправляется на проверку другому члену команды или руководителю.

### • Completed (Завершено)

Описание: Задача полностью выполнена.

Применение: После того, как задача выполнена и проверена, она переводится в состояние "Completed".

### • Closed (Закрыто)

Описание: Задача закрыта, работа над ней полностью завершена и она больше не активна.

Применение: Задача переводится в состояние "Closed" после того, как она была завершена и больше не требует внимания.

## Правила создания новой задачи

Любой участник команды может создавать задачи.

Задачи создаются для организации работы над проектом, фиксации багов или разработки нового функционала.

## Правила перехода задачи из состояния в состояние

Задачи следуют линейной последовательности от "New" до "Closed", но некоторые состояния могут опускаться (например, "Paused" или "Review"), если они не применимы к конкретной задаче.

Повторение состояний возможно в случае возвращения к задаче после паузы или повторной проверки.

# Требования к проекту

Программный продукт "To-Do List" предназначен для управления задачами. Он позволяет пользователям создавать, редактировать, удалять и просматривать свои задачи, а также предоставляет функции для улучшения организации и производительности.

Программный продукт To-Do List состоит из следующих подсистем:

1. Пользовательский интерфейс
2. Подсистема Аутентификации и Регистрации
3. Подсистема Создания Задач
4. Подсистема ввода-вывода
5. Подсистема Редактирования и Удаления Задач
6. Подсистема Отображения Задач
7. Подсистема Статистики
8. Подсистема Уведомлений и Напоминаний
9. Подсистема Импорта/Экспорта Данных

## Требования к подсистеме «Пользовательский интерфейс»

Требование REQ\_UI\_001: Должна быть функция сохранения данных, доступная через кнопку «Сохранить», элемент меню «Файл» и сочетание клавиш «Ctrl + S». Изображено на рисунке 2.

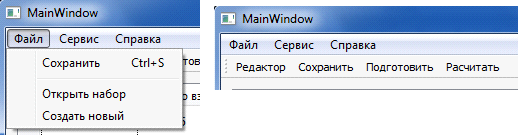


Рисунок 2 – Требования к подсистеме «Пользовательский интерфейс»

Требование REQ\_UI\_002: Система должна создавать новую строку для ввода данных, только если все обязательные поля заполнены.

## Требования к подсистеме «Подсистема аутентификации и регистрации»

Требование REQ\_АР\_001: Надежная аутентификация и регистрация пользователей, включая восстановление доступа.

Надежная: Обеспечение безопасности данных пользователя, использование сильных паролей, многофакторной аутентификации, и защита от несанкционированного доступа.

## Требования к подсистеме «Подсистема создания задач»

Требование REQ\_MCZ\_001: Ввод ключевых данных задачи (заголовок, описание, дата, время, приоритет).

Требование REQ\_MCZ\_002: Валидация данных с обратной связью при ошибках.

Требование REQ\_MCZ\_003: Выбор дополнительных параметров задачи (продолжительность, повторяемость, метки).

## Требования к подсистеме «Подсистема Ввода-Вывода»

Требование REQ\_PVV\_001: Интеграция с внешними источниками для импорта задач.

Требование REQ\_PVV\_002: Экспорт данных в различные форматы для совместимости с другими системами.

Требование REQ\_PVV\_003: Поддержка разных методов ввода данных (импорт из файла, например json, excel).

## Требования к подсистеме «Подсистема Редактирования и Удаления Задач»

Требование REQ\_MRUZ\_001: Интуитивный интерфейс для редактирования всех аспектов задачи.

Требование REQ\_MRUZ\_002: Безопасное удаление задач с возможностью восстановления.

Требование REQ\_MRUZ\_003: Автоматическое обновление связанных задач при изменении деталей.

## Требования к подсистеме «Подсистема Отображения Задач»

Требование REQ\_OP\_001: Отображение задач с фильтрацией (дата, приоритет, статус).

Требование REQ\_OP\_002: Различные режимы просмотра (список, календарь, матрица Эйзенхауэра).

Требование REQ\_OP\_003: Персонализация вида отображения задач (цвета, шрифты, макет).

## Требования к подсистеме «Подсистема Статистики»

Требование REQ\_MS\_001: Генерация диаграмм и графиков для статистики задач.

Требование REQ\_MS\_002: Возможность экспорта статистических данных в различные форматы.

Требование REQ\_MS\_003: Настраиваемые отчеты по выполнению задач.

Требование REQ\_MS\_004: Экспорт статистических данных (возможность передачи данных ввода-вывода для экспорта в разных форматах).

## Требования к подсистеме «Подсистема Уведомлений и Напоминаний»

Требование REQ\_UN\_001: Автоматическая отправка уведомлений о задачах через разные каналы.

Требование REQ\_UN\_002: Настраиваемые напоминания о задачах с выбором времени и частоты.

Требование REQ\_UN\_003: Интеграция с календарями для уведомлений о задачах.

## Требования к подсистеме «Подсистема импорта/экспорта данных»

Требование REQ\_IE\_001: Импорт задач из внешних источников (CSV, JSON, Excel).

Требование REQ\_IE\_002: Экспорт списков задач и статистики в разные форматы.

Требование REQ\_IE\_003: Безопасная передача данных с шифрованием при импорте/экспорте.

# Архитектура проекта

Архитектура программного обеспечения относится к фундаментальным структурам программной системы и дисциплине создания таких структур и систем. Каждая структура включает элементы программного обеспечения, отношения между ними, а также свойства как элементов, так и отношений.

Архитектура программной системы – это метафора, аналогичная архитектуре здания. Он функционирует как план для системы и проекта разработки, в котором излагаются задачи, которые должны быть выполнены командами разработчиков.

## Архитектурно-контекстная диаграмма

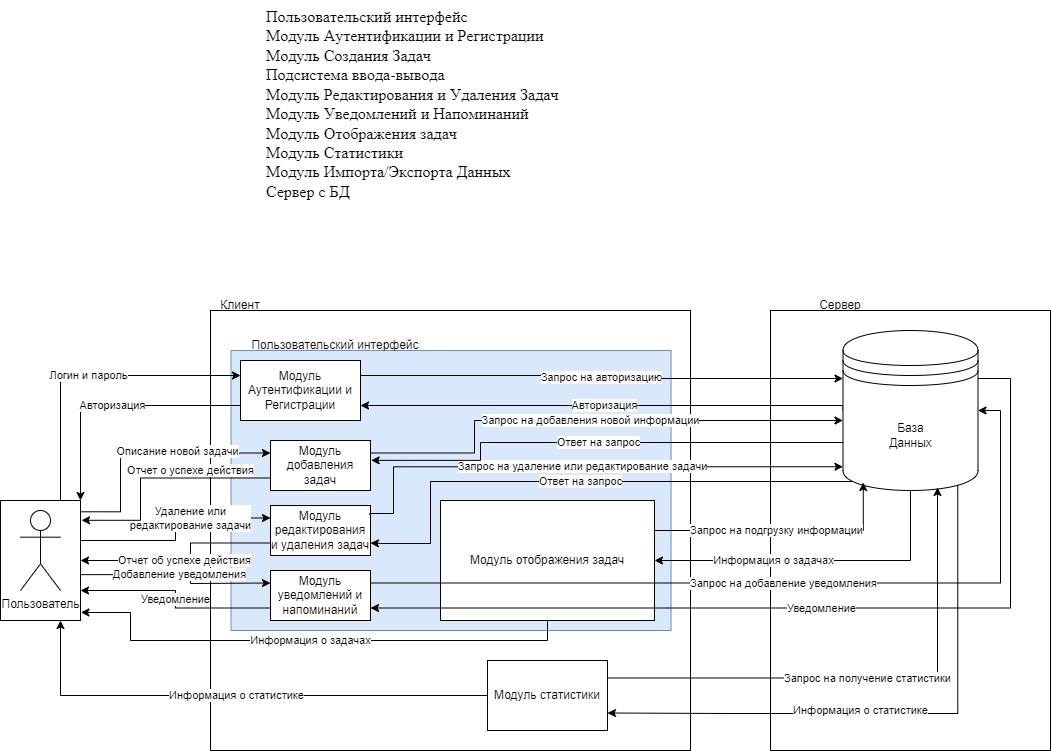


Рисунок 3 – Архитектурно-контекстная диаграмма

## Диаграмма классов

Диаграмма классов в нашем проекте представлена в виде паттерна MVC. MVC, или Model-View-Controller, представляет собой архитектурный паттерн, используемый в разработке программного обеспечения. Этот паттерн разделяет приложение на три основных компонента: Model, View, Controller. Основной идеей MVC является разделение ответственности между этими тремя компонентами, что облегчает управление кодом, повышает его модульность и облегчает тестирование.

View (Представление) - пользователь взаимодействует с интерфейсом. View передает действия пользователя контроллеру.

Controller (Контроллер) - получает информацию от View, обрабатывает пользовательский ввод и обновляет Model.

Model (Модель) - хранит данные и бизнес-логику. После обновления контроллером, оповещает View для обновления интерфейса.

На рисунке 4 представлена общая картина взаимодействия компонентов системы.

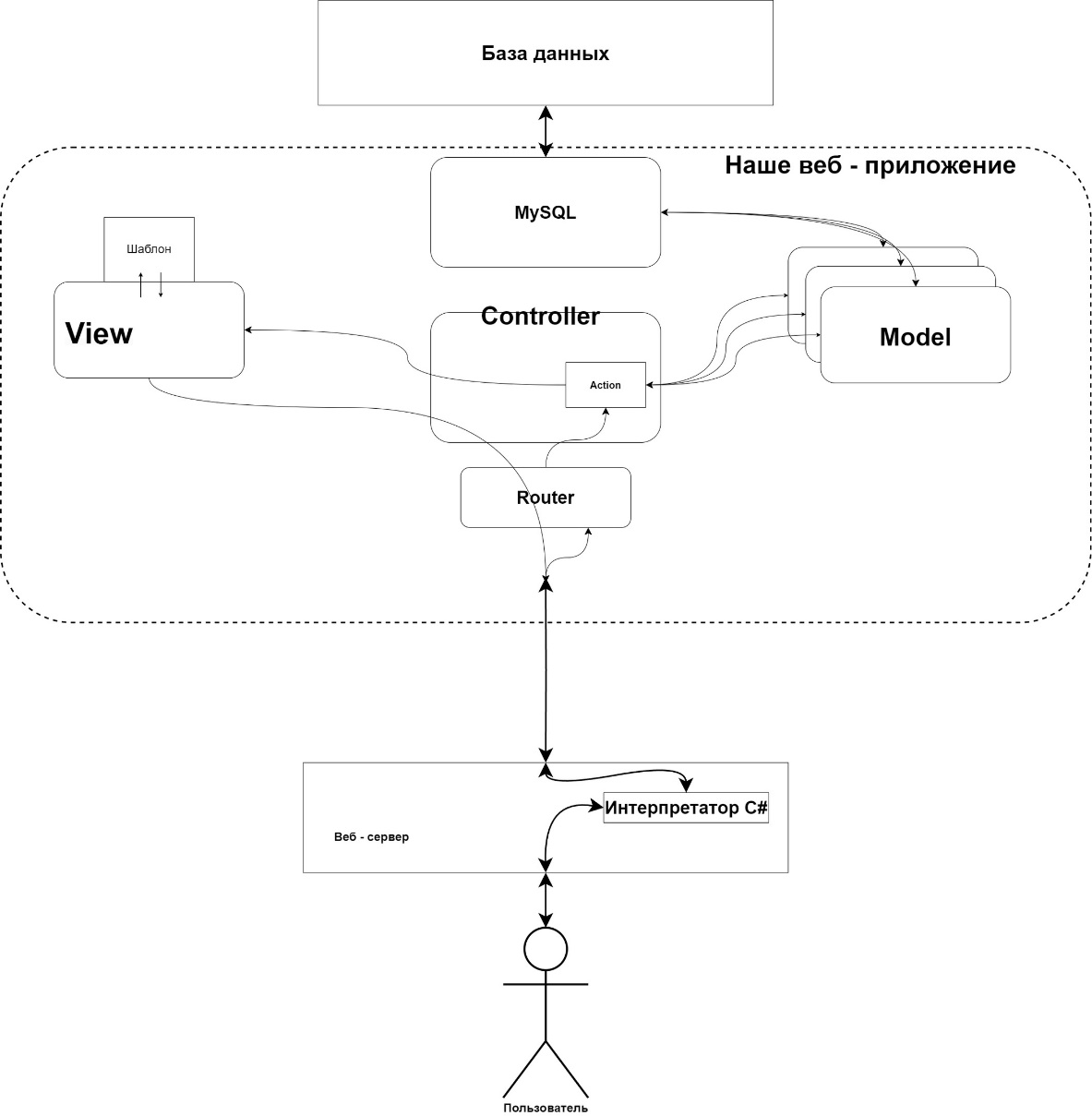


Рисунок 4 – Общая картина взаимодействия компонентов системы

## Классы контроллеры

На рисунке 5 представлена диаграмма классов компонента: «Контроллеры»

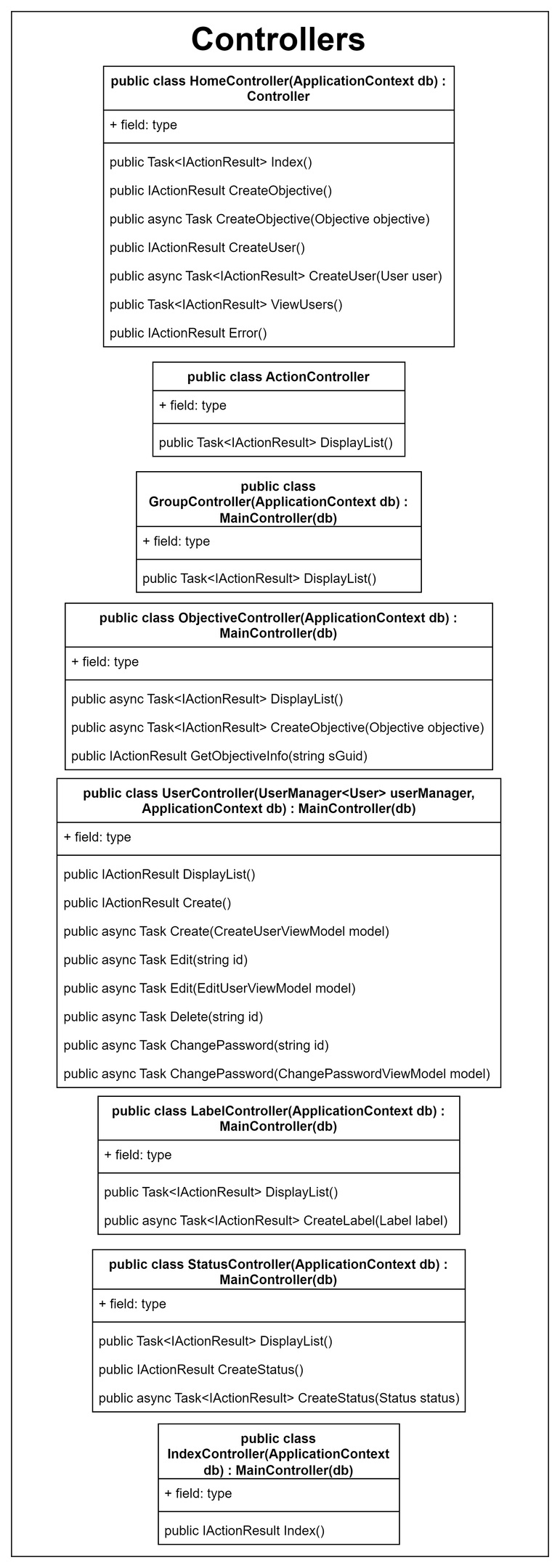


Рисунок 5 – Диаграмма классов компонента «Контроллеры»

## Классы представления

На рисунке 6 представлена диаграмма классов компонента «Представления»

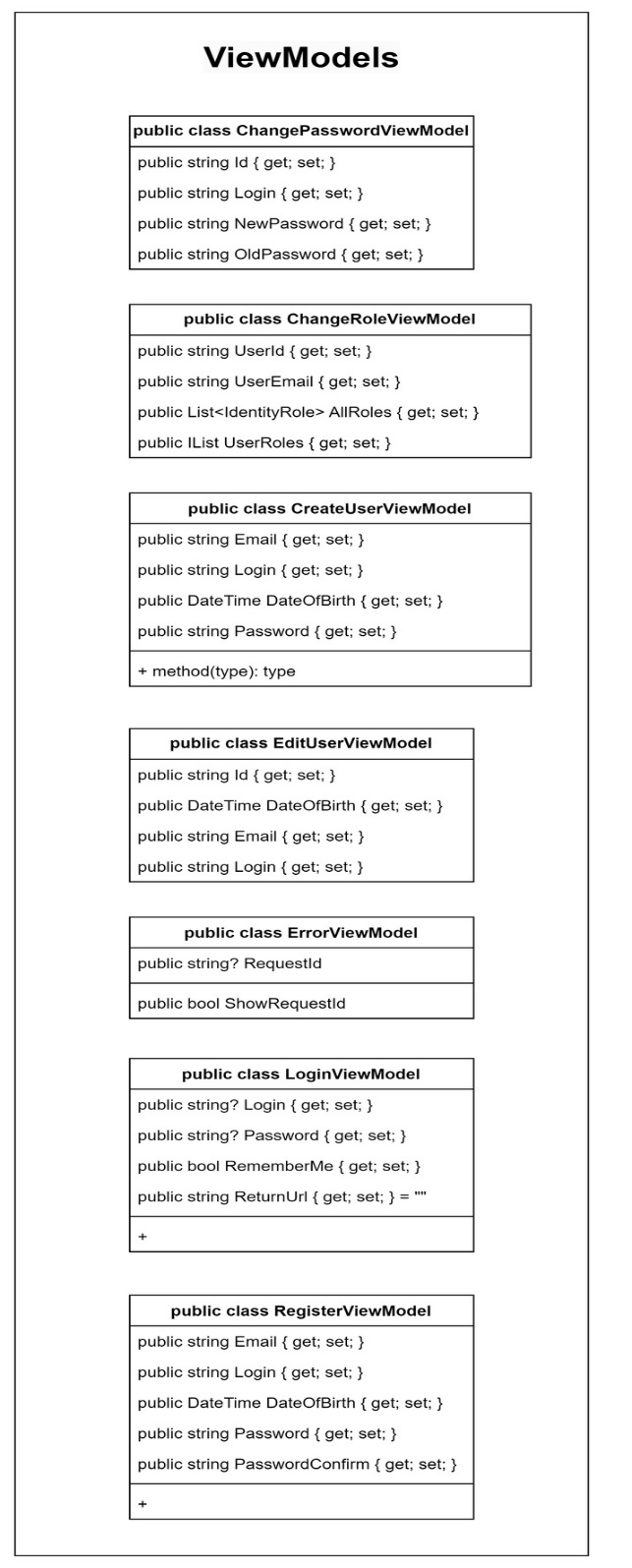


Рисунок 6 – Диаграмма классов компонента «Представления»

## Классы модели

На рисунке 7 представлена диаграмма классов компонента «Модели»

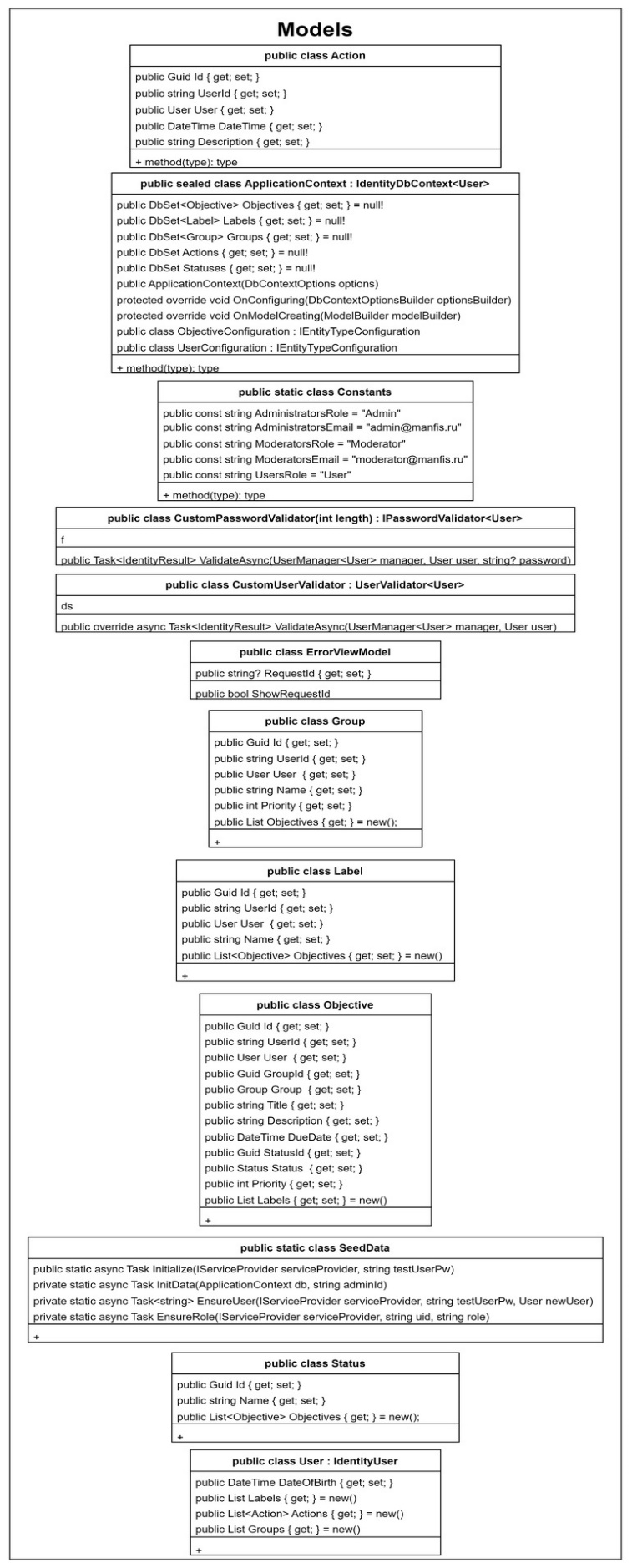


Рисунок 7 – Диаграмма классов компонента «Модели»

# Измерения проекта

Контроль над производственным процессом и его результатами является ключевым видом деятельности на современном предприятии, производящем программное обеспечение на заказ. В силу специфики такого продукта, как программное обеспечение, для оценки эффективности процесса и качества конечного продукта применяются особые методы.

Комплекс мероприятий, направленных на количественную оценку эффективности работы, называется программой измерений.

## Метрика эффективности процесса производства

## IFD (Integrated Function Deployment)

IFD = (Количество найденных ошибок / Размер рабочего продукта)

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – инспекция, измеряемый атрибут – плотность найденных в ходе инспекции ошибок.

Количество найденных ошибок – ошибки, обнаруженные на этапе инспектирования.

Единица измерения – ошибка / <страница, требование**,** LOC, тест>.

## FS (False Scrining)

FS = (Ложные срабатывания / Общее количество срабатываний)

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Ложные срабатывания - количество случаев, когда FS сообщает о наличии ошибок, которые на самом деле отсутствуют.

Общее количество срабатываний - общее количество раз, когда FS выявляет потенциальные проблемы.

Единица измерения – неполадка / LOC.

## IPF (Integrated Product Functionality)

IPF = (Число обнаруженных ошибок до выпуска его релиза) / LOC

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – продукт, измеряемый атрибут – плотность неполадок.

Число обнаруженных ошибок – количество ошибок, обнаруженных на этапе тестирования

Единица измерения – неполадка / LOC.

# Рекомендации по кодированию

При разработке на C# ASP.NET важно придерживаться определенных правил кодирования для обеспечения высокого качества кода, его удобочитаемости и понятности. Эти правила помогают создавать структурированный код и снижать количество ошибок, сохраняя при этом творческую свободу разработчика.

• Соблюдение стиля именования: Использование PascalCase для именования классов и методов, camelCase для переменных и аргументов. Константы следует именовать в UPPER\_CASE.

• Организация пространства имен: Структурирование пространства имен согласно функциональности и логике приложения. Избегание излишне глубокой вложенности.

• Чистота кода: Стремление к написанию чистого и понятного кода. Избегайте длинных методов и классов, разделяйте их на более мелкие и управляемые части.

• Комментарии и документация: Написание понятных комментариев там, где это необходимо, особенно для сложных участков кода. Использование XML-комментарии для документирования классов, методов и их параметров.

• Обработка исключений: Правильная обработка исключений, избегание "пустых" блоков catch. Логирование исключений для упрощения отладки.

• Соблюдение принципов SOLID: Следование принципам SOLID для обеспечения гибкости, масштабируемости и поддерживаемости кода.

• Использование шаблонов проектирования: Применение шаблонов проектирования там, где это уместно, для повышения эффективности и структурированности кода.

• Тестирование: Написание модульных тестов для проверки отдельных компонентов вашего приложения. Стремление к достижению высокого покрытия кода тестами.

• Использование асинхронности: Эффективное использование асинхронные методы для повышения производительности приложения, особенно при работе с I/O операциями.

• Соблюдение конвенций ASP.NET: Следование рекомендациям и лучшим практикам, специфичным для ASP.NET, включая использование MVC, Razor Pages и Entity Framework.

• Рефакторинг и оптимизация: Регулярное проведение рефакторинга для улучшения читаемости и производительности кода. Оптимизирование запросов к базе данных и использование кэширование, где это возможно.

# План тестирования проекта

**Подсистема «Пользовательский интерфейс»**

• TEST\_UI\_1

Тестируемые требования: REQ\_UI\_001

Описание теста: Нажать на кнопку «Сохранить», выбрать опцию «Сохранить» из меню «Файл» и использовать сочетание клавиш «Ctrl + S».

Ожидаемый результат: Данные успешно сохранены в каждом случае.

• TEST\_UI\_2

Тестируемые требования: REQ\_UI\_002

Описание теста: Попытка создать новую строку для ввода данных без заполнения всех обязательных полей.

Ожидаемый результат: Новая строка не создается до тех пор, пока не будут заполнены все обязательные поля.

**Подсистема «Подсистема создания задач»**

• TEST\_MCZ\_1

Тестируемые требования: REQ\_MCZ\_001

Описание теста: Ввести данные задачи, включая заголовок, описание, дату, время и приоритет.

Ожидаемый результат: Задача успешно создана с указанными параметрами.

• TEST\_MCZ\_2

Тестируемые требования: REQ\_MCZ\_002

Описание теста: Ввести некорректные данные в поля задачи и попытаться их сохранить.

Ожидаемый результат: Система предоставляет обратную связь об ошибках, не позволяя сохранить задачу.

• TEST\_MCZ\_3

Тестируемые требования: REQ\_MCZ\_003

Описание теста: Выбрать дополнительные параметры задачи, такие как продолжительность, повторяемость и метки.

Ожидаемый результат: Задача успешно создана с выбранными дополнительными параметрами.

**Подсистема «Подсистема Ввода-Вывода»**

• TEST\_PVV\_1

Тестируемые требования: REQ\_PVV\_001

Описание теста: Импортировать задачи из внешнего источника данных.

Ожидаемый результат: Задачи успешно импортированы в систему.

• TEST\_PVV\_2

Тестируемые требования: REQ\_PVV\_002

Описание теста: Экспортировать данные в различные форматы и проверить их совместимость с другими системами.

Ожидаемый результат: Данные успешно экспортированы и совместимы с другими системами.

• TEST\_PVV\_3

Тестируемые требования: REQ\_PVV\_003

Описание теста: Импортировать данные из файла формата json или excel.

Ожидаемый результат: Данные успешно импортированы в систему.

**Подсистема «Подсистема Статистики»**

• TEST\_MS\_1

Тестируемые требования: REQ\_MS\_001

Описание теста: Генерировать диаграммы и графики для визуализации статистики по задачам.

Ожидаемый результат: Диаграммы и графики корректно отображают статистику задач.

• TEST\_MS\_2

Тестируемые требования: REQ\_MS\_002

Описание теста: Экспортировать статистические данные в различные форматы и проверить их корректность.

Ожидаемый результат: Статистические данные успешно экспортированы и корректны.

• TEST\_MS\_3

Тестируемые требования: REQ\_MS\_003

Описание теста: Сформировать настраиваемый отчет по выполнению задач.

Ожидаемый результат: Отчет соответствует выбранным пользователем параметрам.

**Подсистема «Подсистема Уведомлений и Напоминаний»**

• TEST\_UN\_1

Тестируемые требования: REQ\_UN\_001

Описание теста: Настроить отправку уведомлений о предстоящих и просроченных задачах через различные каналы.

Ожидаемый результат: Уведомления успешно отправлены через выбранные каналы.

• TEST\_UN\_2

Тестируемые требования: REQ\_UN\_002

Описание теста: Настроить напоминания о задачах с выбором времени и частоты.

Ожидаемый результат: Напоминания успешно настроены и работают согласно указанным параметрам.

• TEST\_UN\_3

Тестируемые требования: REQ\_UN\_003

Описание теста: Интегрировать систему уведомлений с календарями пользователей.

Ожидаемый результат: Уведомления успешно интегрированы с календарями пользователей.

# Интерфейс прототипа

## Вид для не авторизированного пользователя

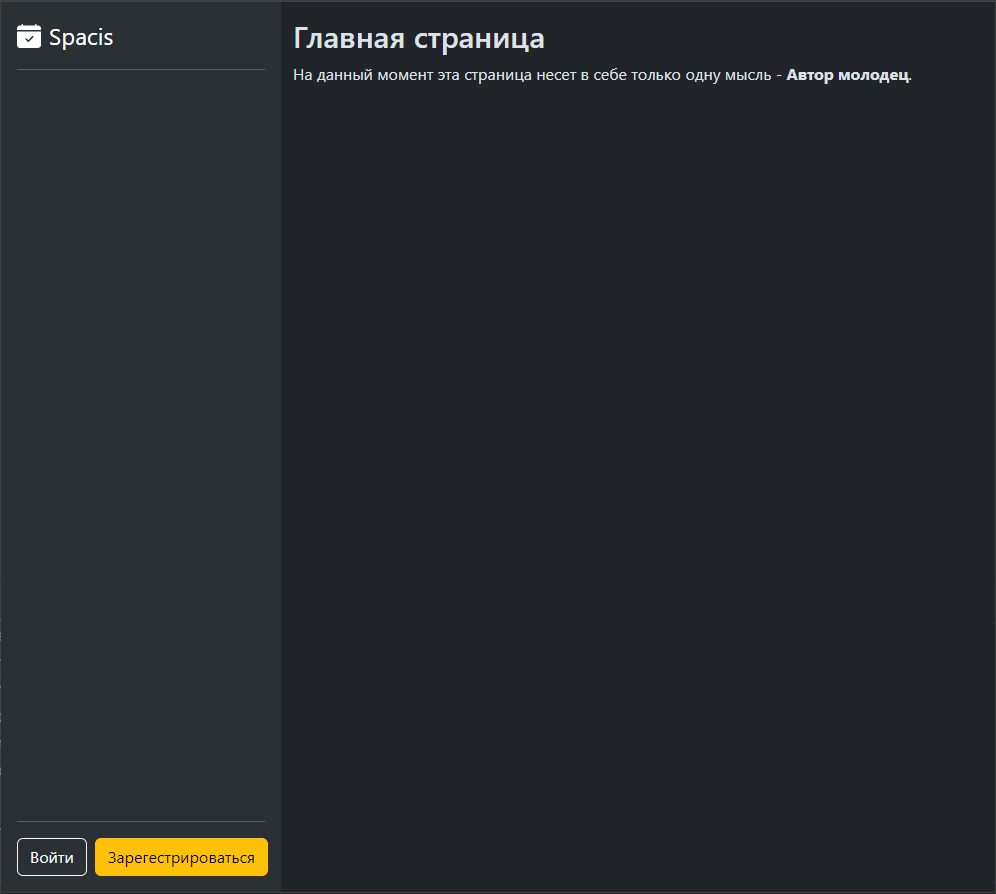


Рисунок 8 – Главная страница

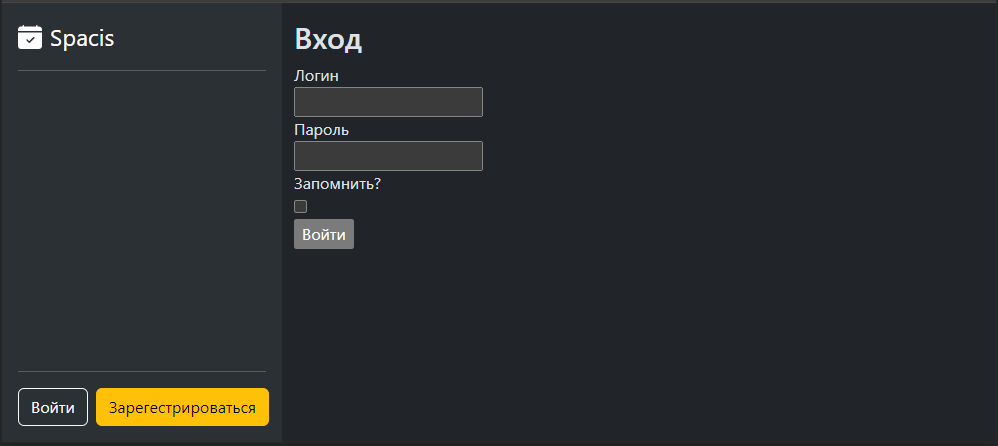


Рисунок 9 – Страница входа

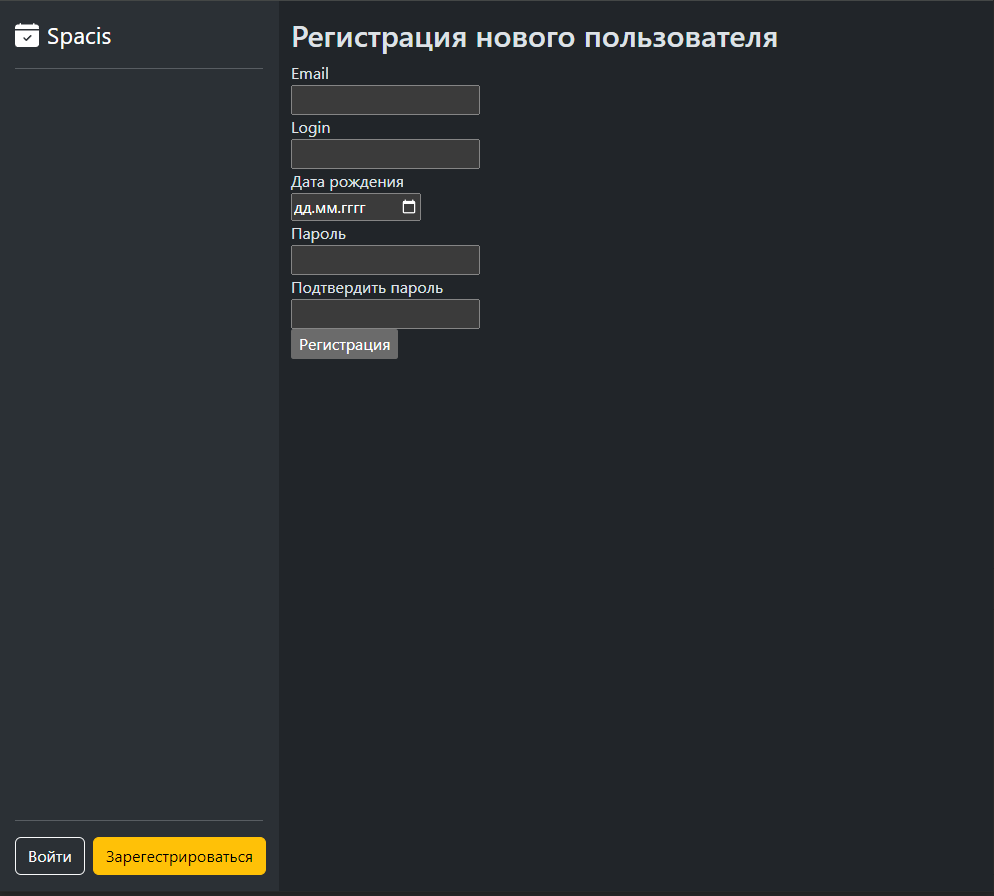


Рисунок 10 – Страница регистрации

## Вид для авторизированного пользователя

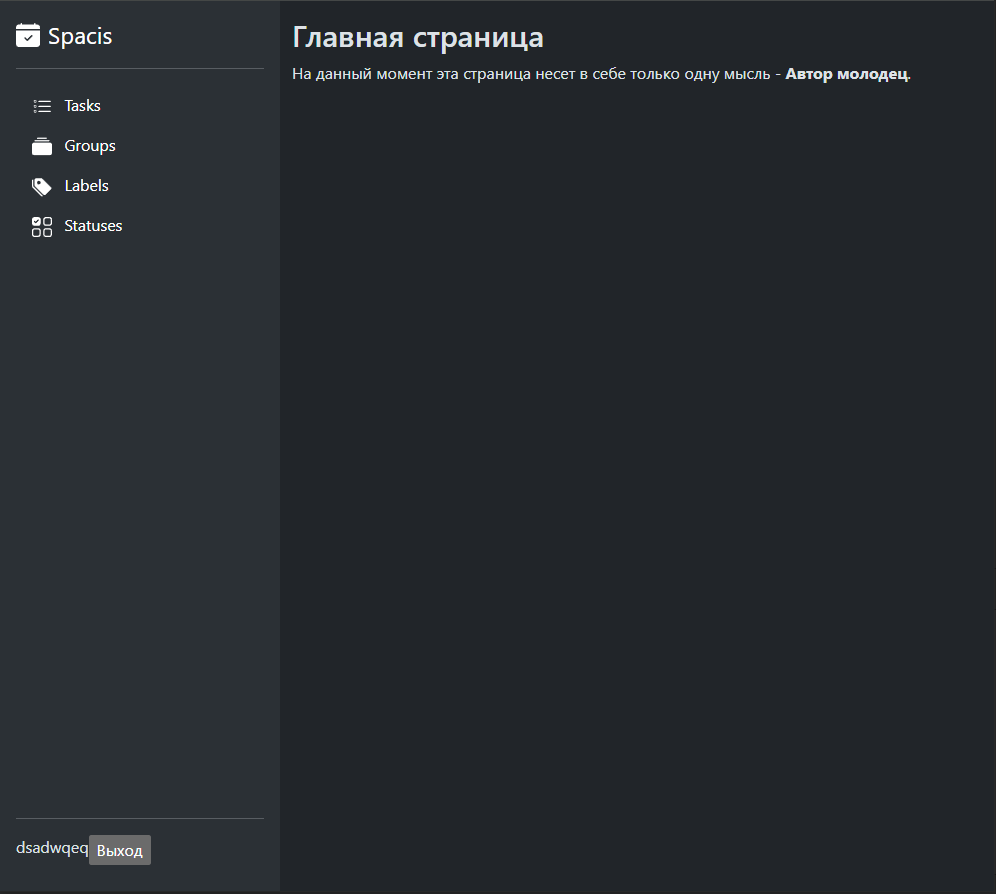


Рисунок 11 – Главная страница

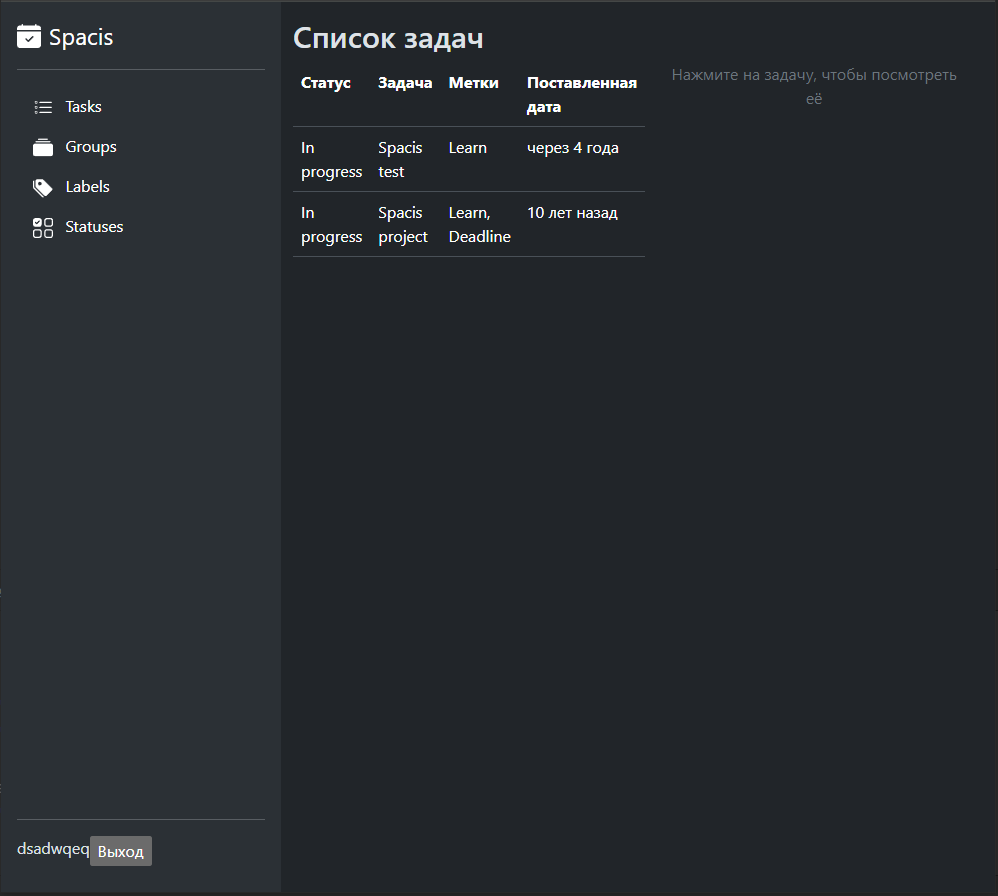


Рисунок 12 – Список задач

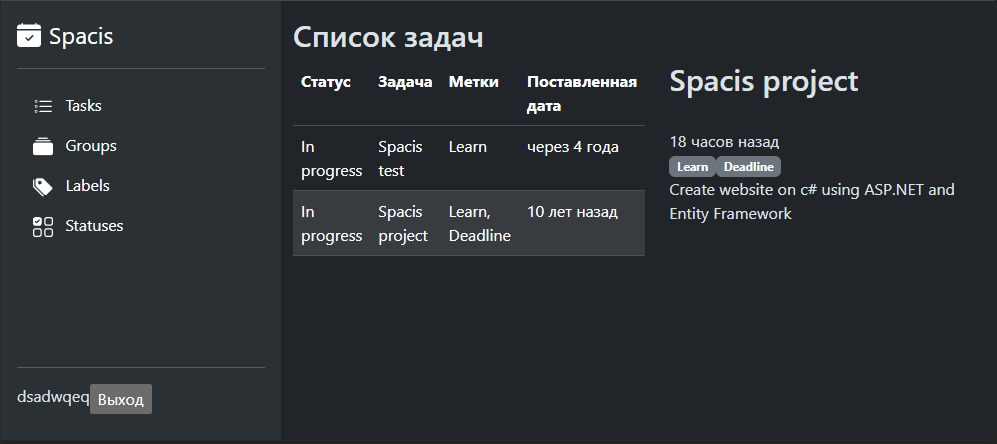


Рисунок 13 – Просмотр определенной задачи

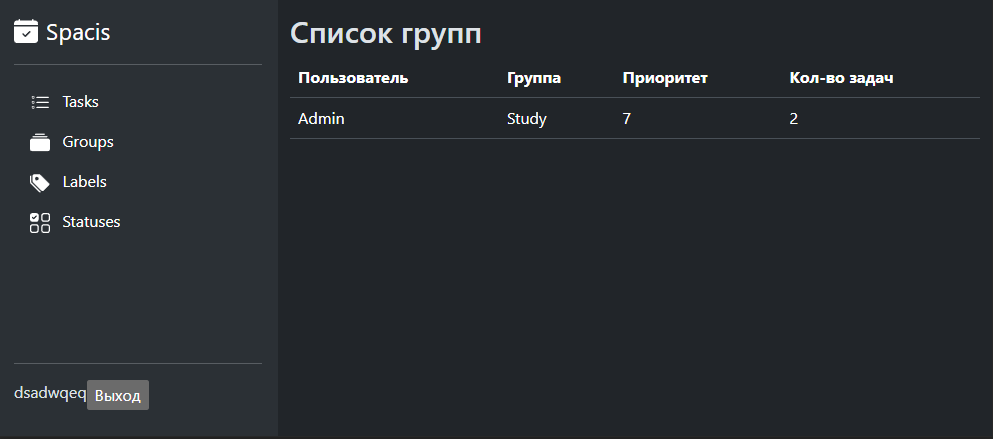


Рисунок 14 – Список групп

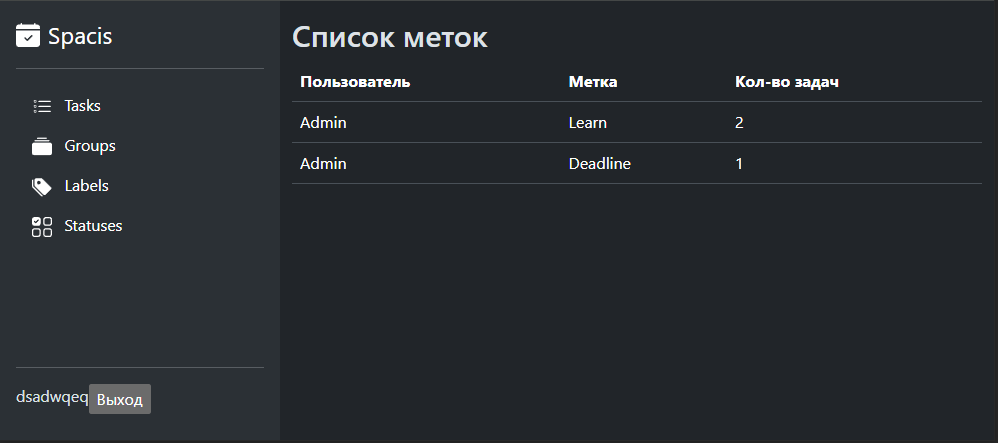


Рисунок 15 – Список меток

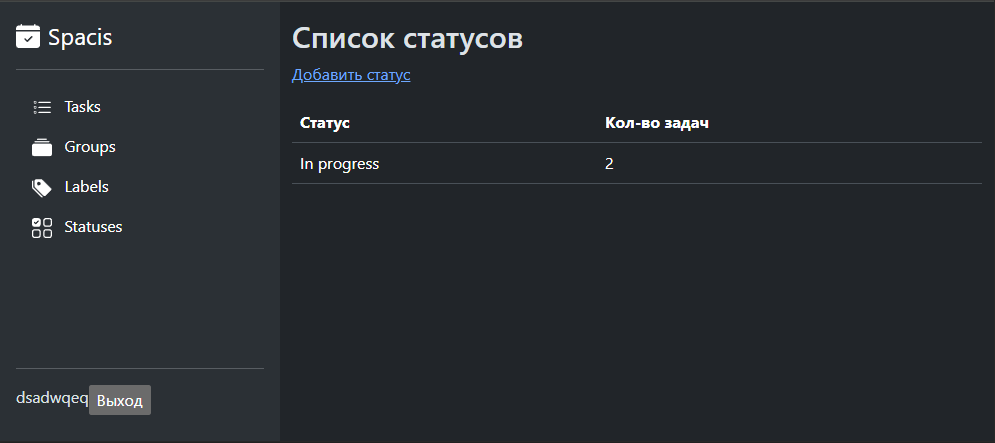


Рисунок 16 – Список статусов

## Вид для привилегированного пользователя

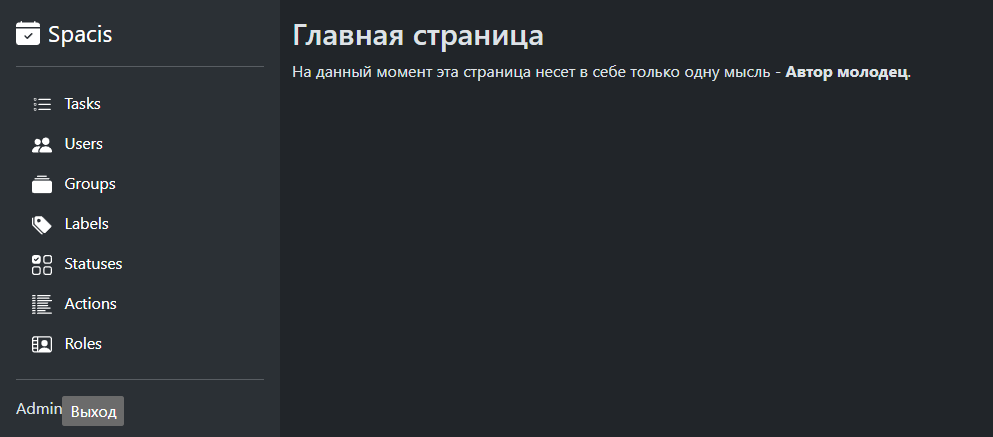


Рисунок 17 – Главная страница

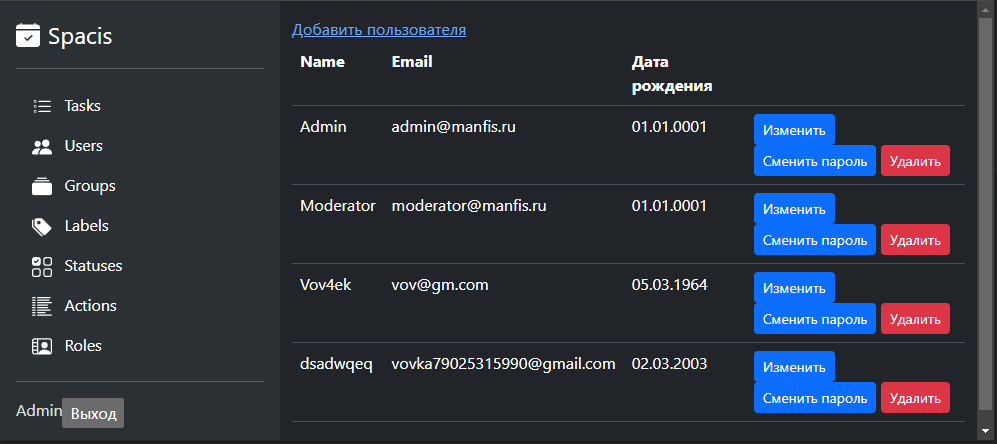


Рисунок 18 – Список пользователей

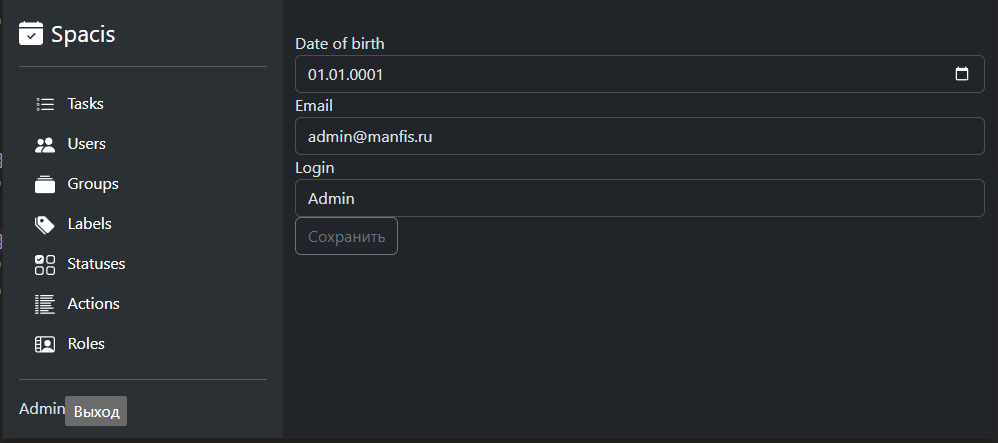


Рисунок 19 – Изменение данных пользователя

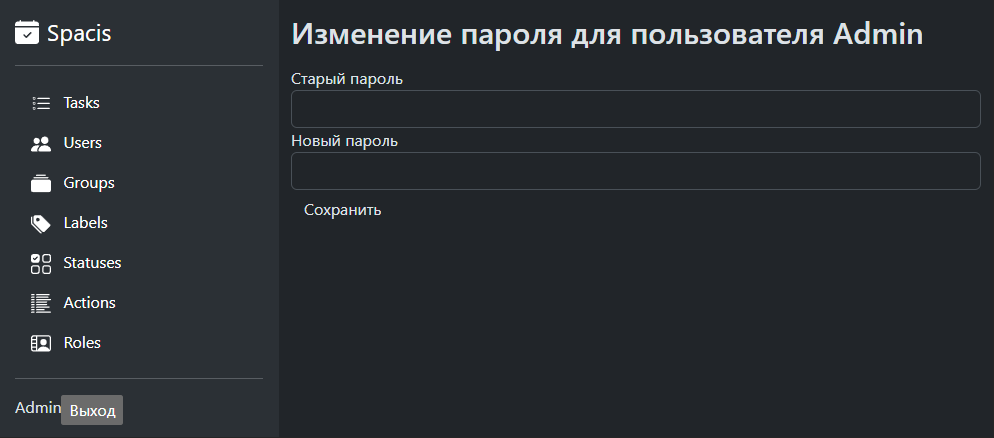


Рисунок 20 – Изменение пароля для пользователя

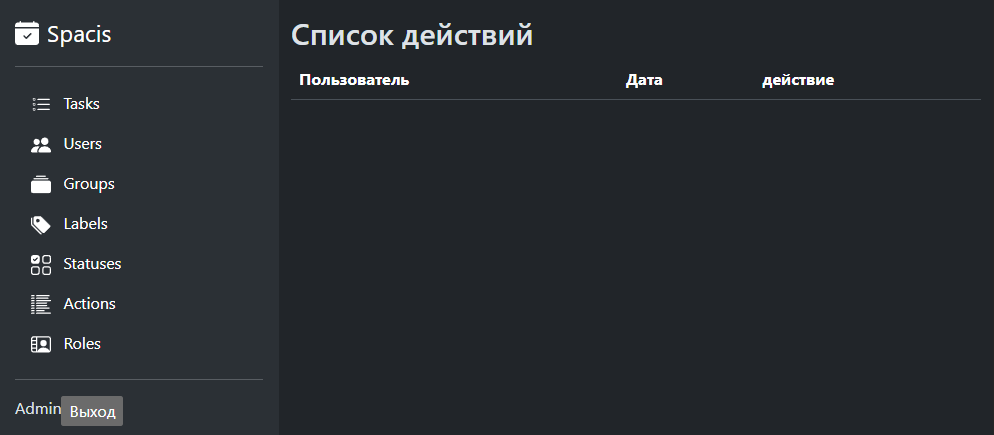


Рисунок 21 – Список действий пользователя

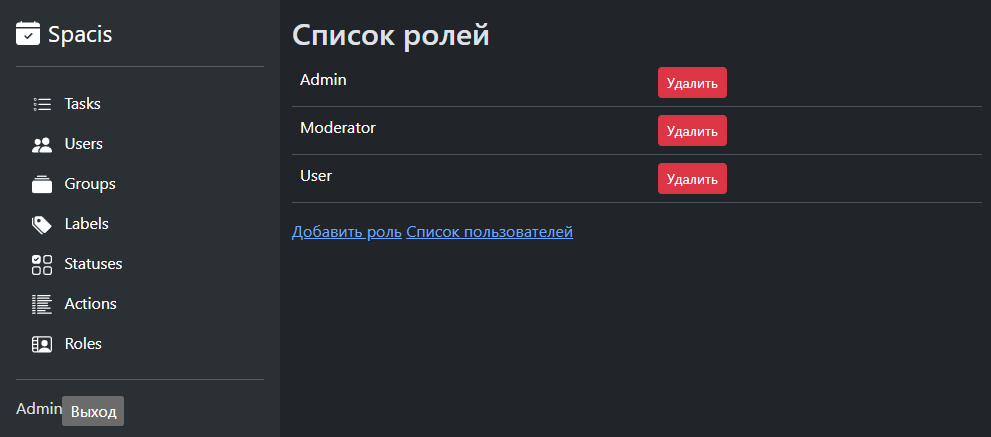


Рисунок 22 – Список ролей

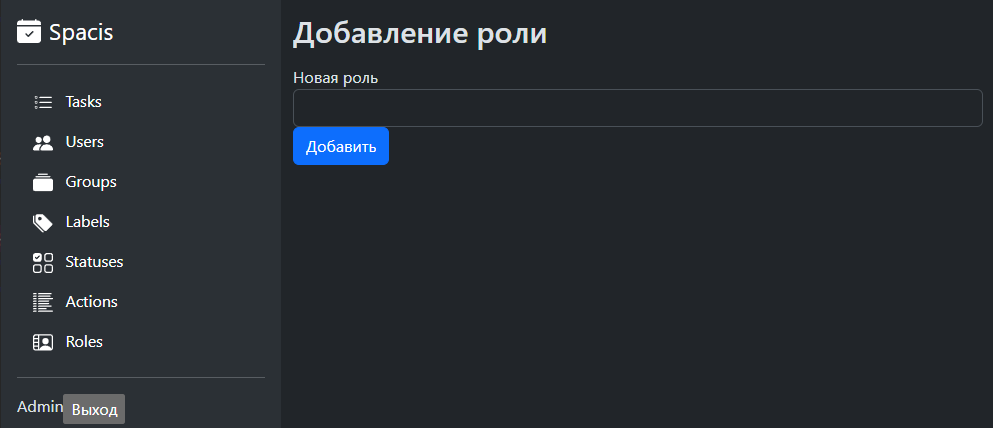


Рисунок 23 – Добавление роли

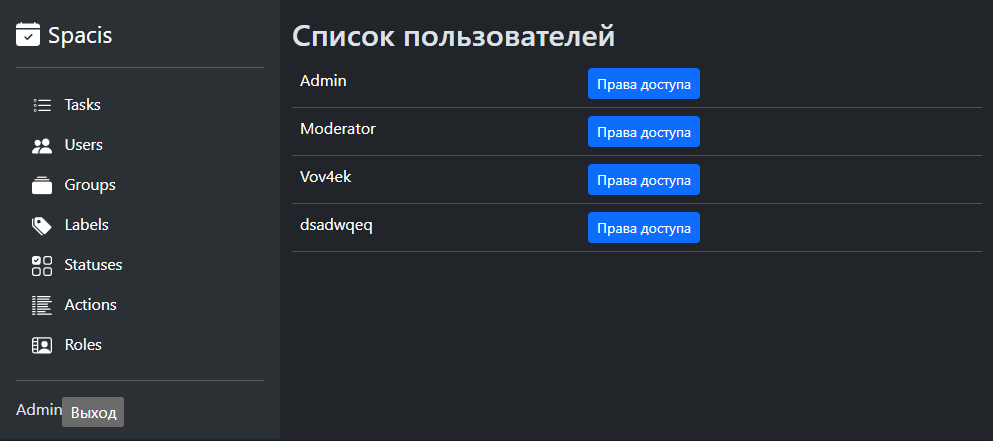


Рисунок 24 – Список пользователей

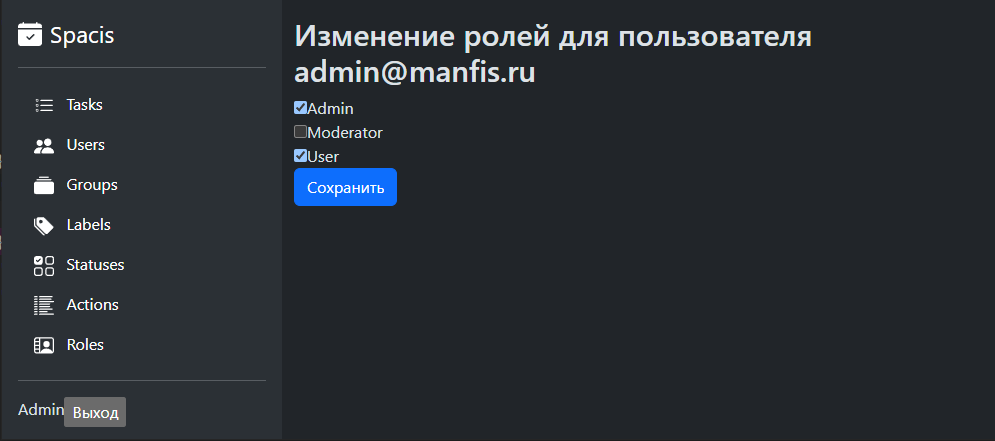


Рисунок 25 – Изменение ролей для пользователя

# Заключение

В рамках курсовой работы было разработано программное средство «To-Do List» с использованием подходов коллективной промышленной разработки, для чего были решены следующие поставленные задачи:

1. Разработан план проекта.
2. Разработан регламент проведения инспекции.
3. Разработана модель состояний задач.
4. Разработаны требования к проекту.
5. Разработана архитектура проекта.
6. Разработаны измерения проекта.
7. Разработаны рекомендации по кодированию.
8. Разработан план тестирования проекта.
9. Разработан интерфейс проекта

Таким образом, цель данной курсовой работы была достигнута.

По окончанию всех работ посчитаем метрики, описанные в 6 главе. Эффективность процесса разработки программного средства – 10 дней работы. Качество продукта выразилось в 15% плотности неполадок.

# Список литературы

1. Гриняк В.М. Лекции по дисциплине «Технологии коллективной промышленной разработки информационных систем». Электронный вариант.
2. Фримен А., Сандерсон А. "ASP.NET Core MVC". - Перевод с английского. - Санкт-Петербург: Питер, 2020. - 832 с.
3. «Официальная документация ASP.NET» - <https://learn.microsoft.com/>. Дата обращения: 15.01.2024