Содержание

[Аннотация 6](#_Toc74663522)

[Введение 7](#_Toc74663523)

[1 Анализ объекта автоматизации 8](#_Toc74663524)

[1.1 Анализ основных бизнес-процессов кафедры 8](#_Toc74663525)

[1.2 Методика формирования рейтинга оценки Webometrics 10](#_Toc74663526)

[1.3 Программные средства статистической обработки данных 13](#_Toc74663527)

[1.4 Методы анализа нелинейных статистических связей 14](#_Toc74663528)

[1.5 Коэффициент детерминации и его использование при оценивании нелинейных статистических связей 15](#_Toc74663529)

[1.6 Постановка задачи 17](#_Toc74663530)

[2 Разработка математической модели для прогнозирования рейтинга Белорусско-Российского университета 19](#_Toc74663531)

[2.1 Исследование рейтинга Белорусско-Российского университета среди вузов Беларуси 19](#_Toc74663532)

[2.2 Методика исследования публикационной активности вуза 21](#_Toc74663533)

[2.3 Анализ вклада IT-кафедр в рейтинг университетов Беларуси 23](#_Toc74663534)

[2.4 Предварительные исследования по построению модели для прогнозирования рейтинга университета 26](#_Toc74663535)

[2.5 Исследование адекватности регрессионной модели 30](#_Toc74663536)

[3 Программная реализация информационной системы 33](#_Toc74663537)

[3.1 Выбор программных средств реализации ИС 33](#_Toc74663538)

[3.2 Структура программного обеспечения ИС 33](#_Toc74663539)

[3.3 Разработка программного кода ИС 34](#_Toc74663540)

[3.4 Руководство пользователя и контрольные примеры 37](#_Toc74663541)

3.5 Рекомендации по повышению рейтинга университета в Webometrics……42

[4 Политика информационной безопасности 44](#_Toc74663542)

[4.1 Цель, принципы и задачи защиты информации в информационной системе 44](#_Toc74663543)

[4.2 Методы и средства обеспечения защиты информационных ресурсов 45](#_Toc74663544)

[4.3 Средства защиты информации и информационных ресурсов 45](#_Toc74663545)

[5 Организационно-экономическая часть 46](#_Toc74663546)

[5.1 Общая постановка к технико-экономическому обоснованию 46](#_Toc74663547)

[5.2 Расчет трудоемкости (производительности) 48](#_Toc74663548)

[5.3 Расчет единовременных затрат (инвестиций) 49](#_Toc74663549)

[5.4 Расчет годовых текущих издержек 57](#_Toc74663551)

[5.5 Расчет показателей экономической эффективности 64](#_Toc74663552)

[5.6 Организация внедрения системы 65](#_Toc74663553)

[5.7 Заключение по разделу 66](#_Toc74663554)

[6 Охрана труда 68](#_Toc74663555)

[6.1 Система управления охраной труда на предприятии 68](#_Toc74663556)

[6.2 Оценка условий труда по показателям напряженности трудового процесса на рабочем месте 70](#_Toc74663557)

[6.3 Выводы и предложения 73](#_Toc74663558)

[7 Энергосбережение и ресурсосбережение при внедрении новых технологий.. ………………………………………………………………………………….....74](#_Toc74663559)

[Заключение 77](#_Toc74663560)

[Список использованных источников 78](#_Toc74663561)

[Приложение А 80](#_Toc74663562)

**Аннотация**

на дипломный проект

Информационная система прогнозирования рейтинга университета

Структура и объем проекта

Дипломный проект состоит из 7 листов графической части и пояснительной записки на 81 страницу. Пояснительная записка состоит из задания, аннотации, введения, семи глав, заключения, списка литературы.

Проект содержит 26 иллюстраций и 17 таблиц. Список литературы включает 18 наименований.

Содержание проекта

В введении определена цель проекта и его актуальность.

В первой главе анализ объекта автоматизации.

Во второй главе описывается разработка математической модели для прогнозирования рейтинга.

В третьей главе описывается программная реализация разработанной информационной системы.

В четвертой главе рассмотрена политика информационной безопасности.

В пятой главе рассмотрена организационно-экономическая часть.

В шестой главе рассмотрены вопросы охраны труда.

В седьмой главе рассмотрены вопросы энергосбережения.

В заключении производится анализ степени выполнения, приводятся итоги проделанного исследования.

**Введение +**

Предлагаю подумать над этим вконце, как и над постановкой задачи **1 Анализ объекта автоматизации**

* 1. **Анализ процесса разработки программного обеспечения +**

Зачастую при разработке программного обеспечения члены команды сталкиваются со многими проблемами, которые необходимо решать для развития проекта. Например, во время создания очередной задачи нужно продумать: какие данные вносить для достаточно подробного описания задачи, в каком формате и по какому шаблону это делать. Ведь очень важно придерживаться некоторого общего шаблона, чтобы после создания задачи одним разработчиком, второму не пришлось собирать воедино всё, что сделал первый. Так же членам команды нужно придумать, как делиться новыми задачами, как обсуждать эти задачи, чтобы о результатах обсуждения могли узнать остальные разработчики. Само собой разумеется, что всем членам команды необходимо иметь возможность контролировать состояние каждой задачи в каждый момент времени, чтобы не начать работать над одной и той же задачей одновременно. При каком бы то ни было взаимодействии, важно понимать, какая роль у вашего собеседника. Часто приходится тратить время на то, чтобы найти человека с конкретной ролью на проекте: тестировщик, разработчик, менеджер и т. д. Очевидно, что всем участникам проекта необходимо обезопасить свою среду взаимодействия и результаты работы от внешних угроз. Процесс выполнения одной технической задачи во время разработки программного обеспечения можно представить функциональной моделью рисунок 1.1.

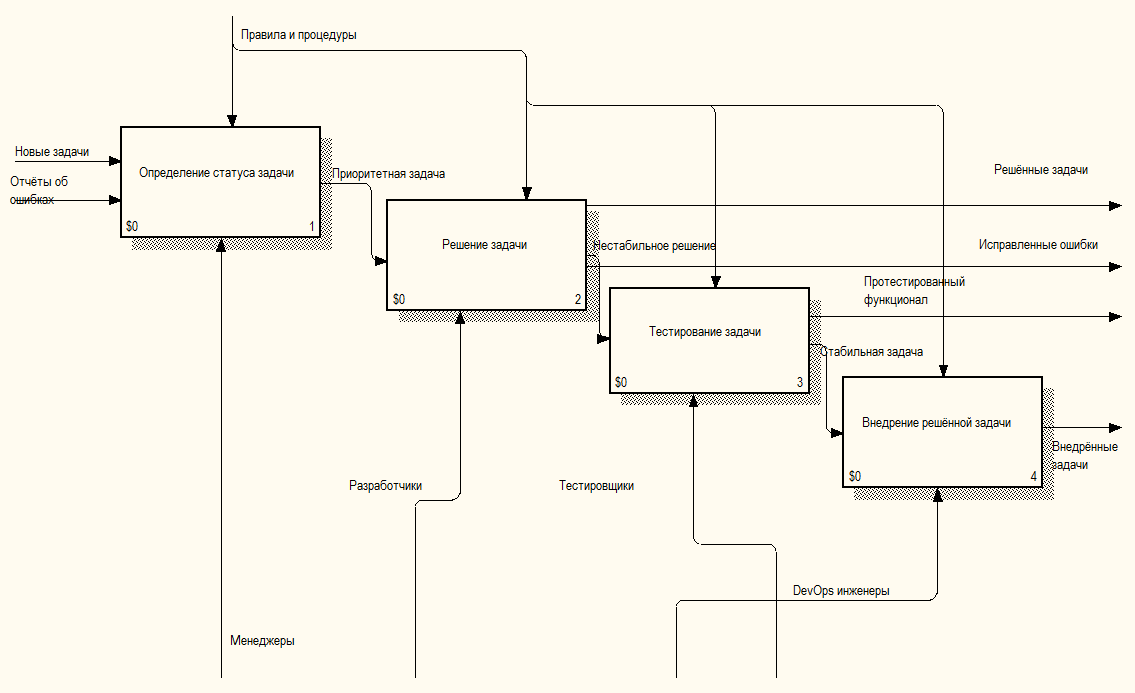


Рисунок 1.1 – Декомпозиция процесса «Выполнение технической  
задачи»

Как видно на данном рисунке, выполнение задачи включает в себя определение статуса и непосредственное создание задачи, решение задачи посредством написания кода, тестирование задачи и внедрение законченного решения в продукт. Результатом этих процессов является законченная и работоспособная часть новой или изменённой логики конечного приложения.

Каждый из вышеперечисленных блоков так же может быть декомпозирован на слои рисунки 1.2 – 1.5.

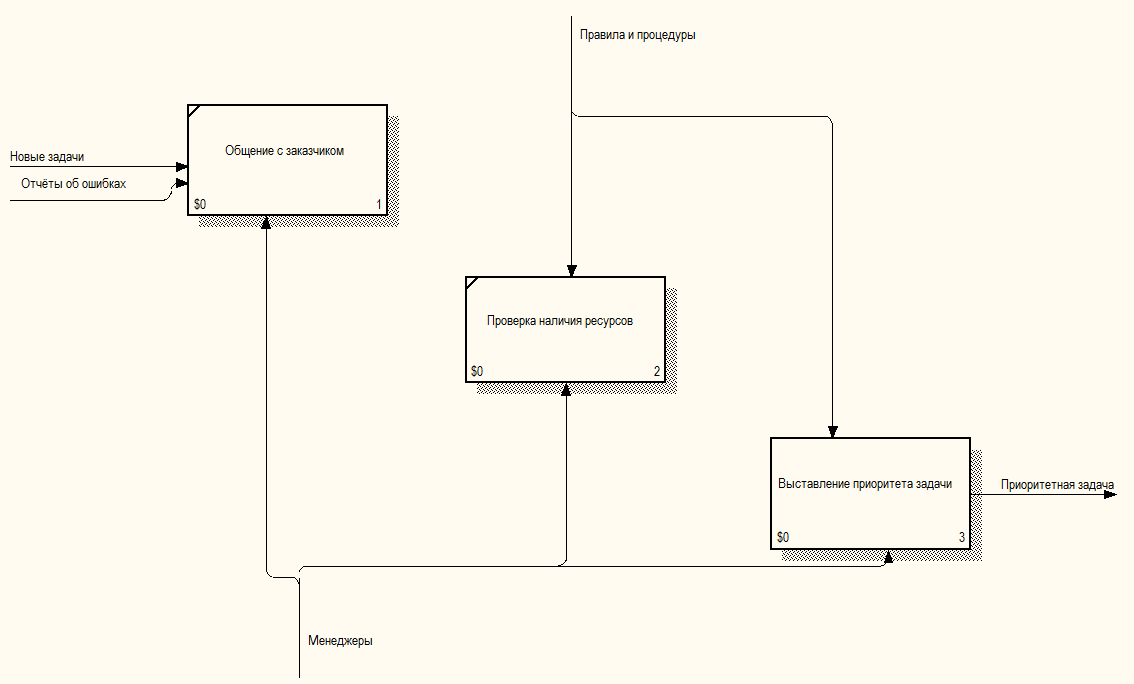


Рисунок 1.2 – Декомпозиция процесса «Определение статуса задачи»

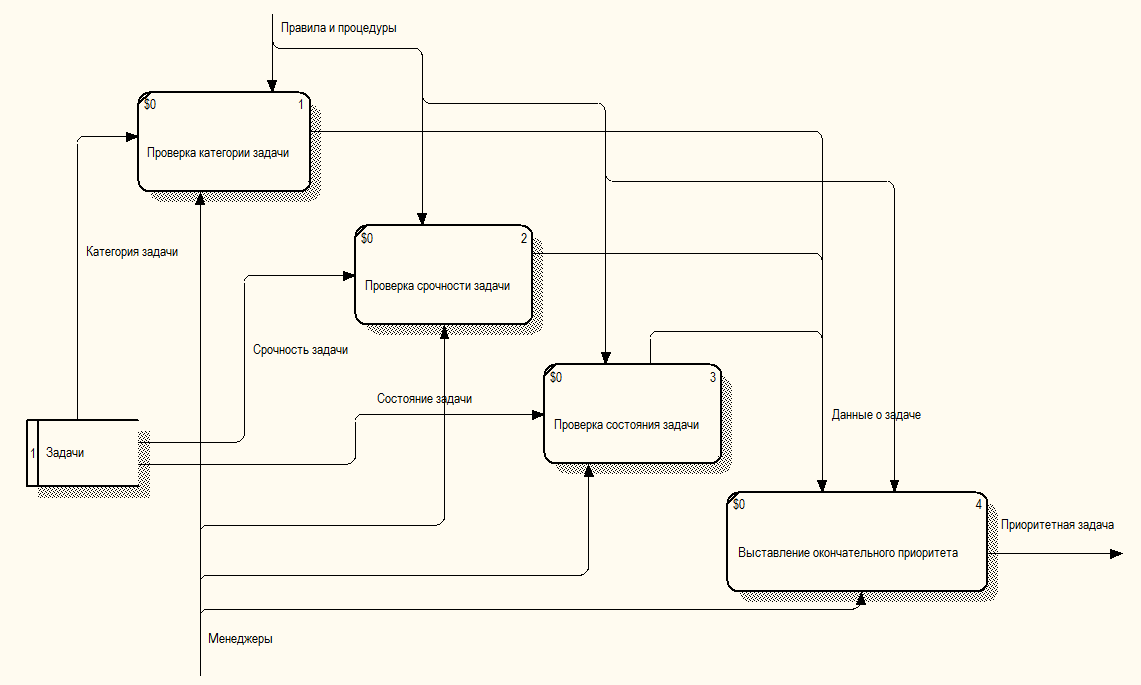


Рисунок 1.3 – Декомпозиция процесса «Решение задачи»

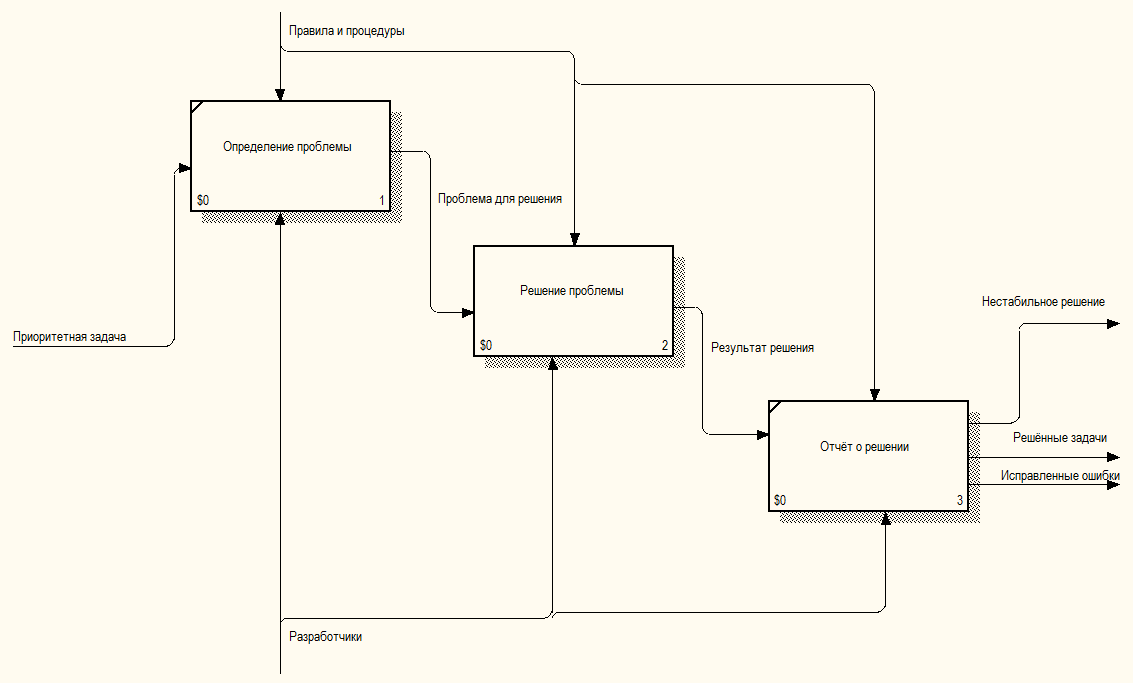


Рисунок 1.4 – Декомпозиция процесса «Тестирование задачи»

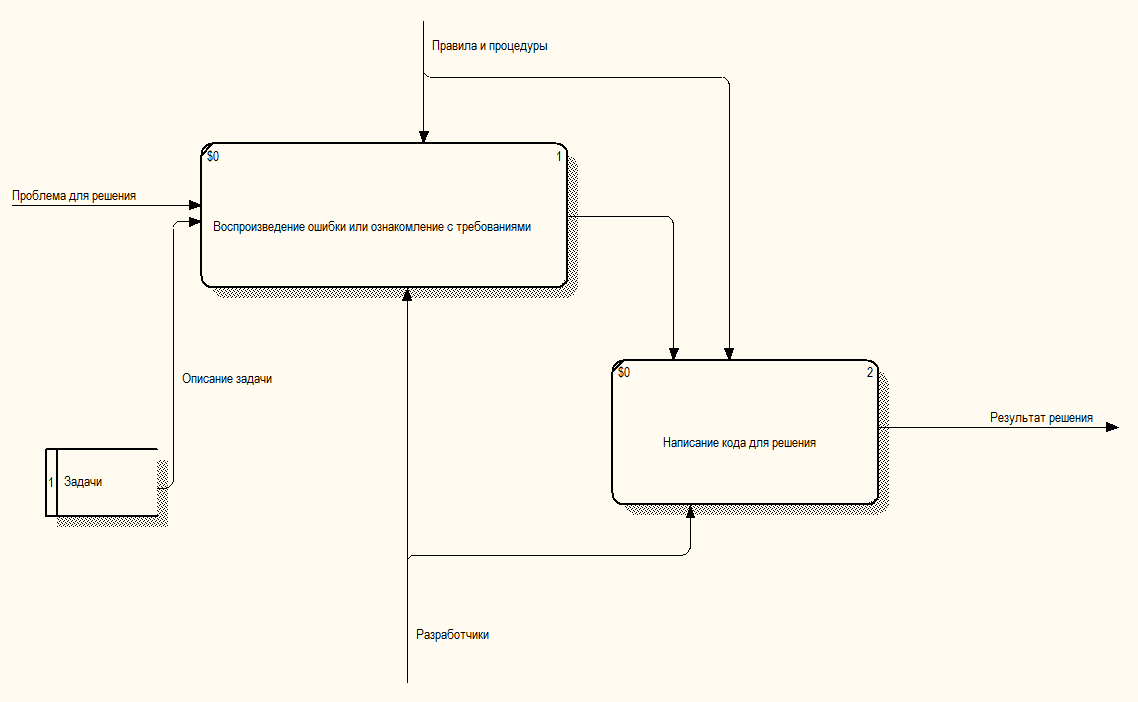


Рисунок 1.5 – Декомпозиция процесса «Внедрение решённой задачи»

Как можно заметить, на каждом из уровней декомпозиции всем членам команды необходимо проделывать множество разнообразных действий. В том числе и по кооперированию друг с другом. Если не прибегать к некоторому дополнительному программному обеспечению, на выполнение всех этих действий уйдёт огромное количество времени. А одна из целей практически любой разработки – это завершение создания продукта за минимальное время. Более того, экономя время на любом виде взаимодействия с командой, каждый отдельно взятый член этой команды может потратить больше времени на повышения качества разрабатываемого продукта. Именно поэтому необходим инструмент, который и позволит эффективнее использовать время команды.

**1.2 Описание проблем и их решение**

Таким образом можно вынести список основных возникающих проблем при разработке ПО, на которые тратиться дополнительное время:

* Обсуждение плана разработки или отдельно взятой задачи.
* Предоставление результатов обсуждений всей команде.
* Описание и объяснение задач
* Контроль задач и сотрудников на проекте
* Защита всей информации на проекте. Контроль доступа к данным.

Данные проблемы можно решить следующим образом:

* Необходим механизм, позволяющий общаться между собой в реальном времени. Результаты общения должны быть видны всем членам команды.
* Необходим механизм оценки трудоёмкости задач.
* Необходим механизм визуализации связей между задачами.
* Необходим механизм полного контроля состояния задач.
* Необходим механизм, для стандартизации всех входных данных.
* Необходим механизм, позволяющий ознакомится с каждым членом команды. Так же должна быть возможность для связи со всеми участниками проекта.
* Необходим механизм авторизации и аутентификации.

Для решение всех вышеперечисленных проблем при разработке серверной части приложения необходимо создать следующие инструменты. Будет внедрён механизм авторизации и аутентификации посредством JWT-токенов и стандартных средств языков программирования. Будет разработана база данных для хранения всех программных сущностей. Будет разработан алгоритм создания графа, описывающего связи между задачами, а также алгоритм определения критического пути на графе с учётом оценок трудоёмкости задач. Будет разработана API для предоставления клиентской части всех необходимых инструментов, таких как доступ к базе данных, взаимодействия с сущностями в ней. Будет разработан механизм разграничения на роли пользователей. Будет внедрён механизм кэширования часто используемых данных.

**1.3** **Программные средства автоматизации процесса взаимодействия команд**

Уже созданные подобные инструменты, такие как Jira и Azure DevOps, зачастую слишком сложны и не подходят для небольших команд разработчиков, не имеющих опыта в координации и разделении задач во время разработки. Наш инструмент, в отличие от вышеперечисленных, проще в освоении и использовании, так как не настолько нагружен функциональными особенностями, которые обычно используются в крупных компаниях или опытными разработчиками при разработке больших приложений.

В Azure, например, при загрузке некоторых компонентов экран просто темнеет. Это сбивает с толку пользователя, заставляя думать, что произошла ошибка. В нашем варианте пользователю показывается специальный загрузочный компонент. Так что пользователь понимает, что идет процесс загрузки данных. Так же в Azure, чтобы добраться до некоторых компонентов приложения, нужно сделать очень много кликов мыши, т.к. эти компоненты имеют глубокую вложенность. В нашем решении все инструменты доступны за один-два клика. У Jira большим недостатком является долгая загрузка всех страниц. Особенно критично это ощущается, когда на каждое новое действие пользователя Jira начинает загружать новую страницу. Так же у Jira по умолчанию настроены ограничения на состояния задач. То есть нельзя свободно перетягивать задачу из одного состояния в другое. Такая возможность появляется только если соблюдены определённые условия, которые могут быть неочевидны пользователю.

Таким образом необходим механизм, который учитывал бы все неудобства, которые существуют в похожих реализованных систем.

**1.4 Математическая модель +**

При одновременной работе нескольких программистов задачи могут выполняться параллельно. В таком случае разные разработчики выполняют разные задачи. Однако они в силу различных причин тратят на выполнение задач неодинаковое количество времени.

Например, рассмотрим следующий граф (рисунок 1). Состояния 1 и 2 являются соответственно начальным и конечным состояниями некоторого разрабатываемого приложения в некотором отрезке времени. Для перехода из состояния 1 в состояние 2 необходимо выполнить все задачи, обозначенные буквами. В процессе параллельной разработки задачи A1, B1-2, C1-2 могут быть выполнены существенно быстрее задач A2, C3. В таком случае разработчики, закончившие раньше, начинают простаивать, что является неэффективным использованием времени.

Наше приложение устраняет причины лишних затрат времени.

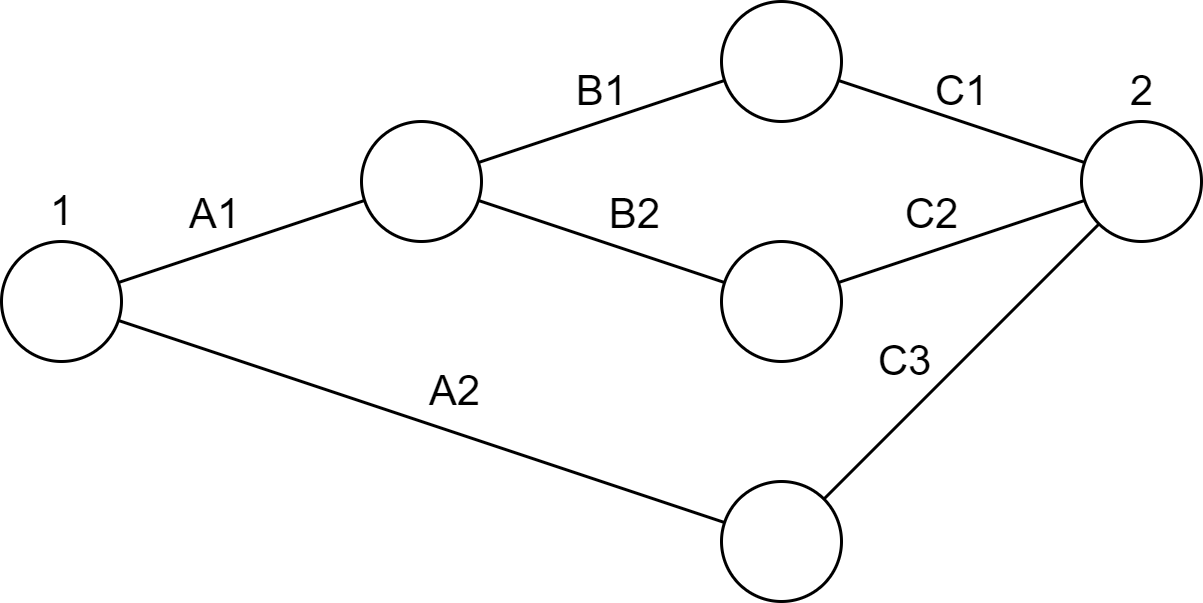


Рисунок 1 — Граф состояний проекта

Для начала введём понятие критического пути. Критический путь состоит из самой длительной последовательности задач от начала проекта до его завершения. Тогда время, которое необходимо затратить на разработку всего проекта (*T*), равно времени, затрачиваемом на каждую задачу, лежащую на критическом пути. Тогда:

где *t* — время, которое необходимо затратить на одну задачу,

*i* — номер задачи на критическом пути.

Каждая задача состоит из одинаковых этапов. Тогда *t* можно представить как:

где *tdes* — время на описание задачи,

*tdev* — время на непосредственно написание кода,

*tcoop* — время, затрачиваемое на взаимодействие членов команды.

Тогда *T* можно представить так:

Из формулы видно, что, уменьшая время на выполнение задач, лежащих на критическом пути, можно достичь сокращения времени на выполнение всего проекта.

**1.5 Постановка задачи**

Придумаем в конце.

**3 Программная реализация информационной системы**

**3.1 Выбор программных средств реализации ИС**

При разработке данной информационной системы в качестве основного средства разработки был выбран язык программирования C#, т.к. он занимает лидирующие позиции среди языков программирования, используемых для написания веб-приложений. C# был использован с платформой .Net 6, т.к. эта платформа обладает всеми необходимыми пакетами для создания гибких и производительный клиент-серверных приложений. Для создание графического интерфейса использовался Java Script в связке с React, который позволяет создавать интерактивные веб-приложения. В качестве базы данных была выбрана MS SQL Server благодаря своим возможностям интеграции с языком C# и многими продуктами компании Microsoft. Приложение уже находится в открытом доступе по ссылке <https://justinmind.azurewebsites.net>. Так же приложение может использоваться локально, для этого необходимо произвести предварительную настройку базы данных и окружения. После этого приложение можно свободно использовать на локальной машине. При некоторых дополнительных доработках есть возможность внедрить приложение на кафедры вузов для контроля учебного процесса. Сфера использования приложения не ограничена конкретными рамками, оно может быть использовано в различных целях.

Во время разработки приложения было использовано большинство современных и эффективных подходов и практик. В частности:

API — Application Programming Interface (интерфейс для программирования приложений). То есть это некоторый интерфейс, позволяющий клиентскому приложению обращаться к серверному приложению. Таких клиентских приложений может быть несколько, и они могут обращаться к одному серверному приложению через общий интерфейс.

REST — Representational State Transfer (передача состояния представления). Это актуальный архитектурный подход для создания API.

Redux — популярный менеджер состояний в веб-приложениях. Обычно его используют в связке с React, но поддержка не ограничена только этой популярной JS-библиотекой

Saga — это библиотека, которая призвана упростить и улучшить побочные эффекты (т.е. такие действия, как асинхронные операции, например, загрузки данных, и "грязные" действия, такие, как доступ к браузерному кешу), сделать лёгкими в тестировании и лучше справляться с ошибками.

· SOLID - это аббревиатура пяти основных принципов проектирования в объектно-ориентированном программировании — Single responsibility, Open-closed, Liskov substitution, Interface segregation и Dependency inversion. В переводе на русский: принципы единственной ответственности, открытости / закрытости, подстановки Барбары Лисков, разделения интерфейса и инверсии зависимостей

· KISS - это принцип проектирования и **программирования**, при котором простота системы декларируется в качестве основной цели или ценности. Есть два варианта расшифровки аббревиатуры: keep it short and simple.

· Паттерн Repository - Репозиторий позволяет абстрагироваться от отдельных подключений к источникам данных, с гарантией работы программы, и является промежуточным звеном между классами, взаимодействующими с данными, и структурами.

· Паттерн Singleton - порождающий паттерн, который гарантирует, что для определенного класса будет создан только один объект, а также предоставит к этому объекту точку доступа.

· Dependency Injection - то стиль настройки объекта, при котором поля объекта задаются внешней сущностью. Другими словами, объекты настраиваются внешними объектами. DI — это альтернатива самонастройке объектов

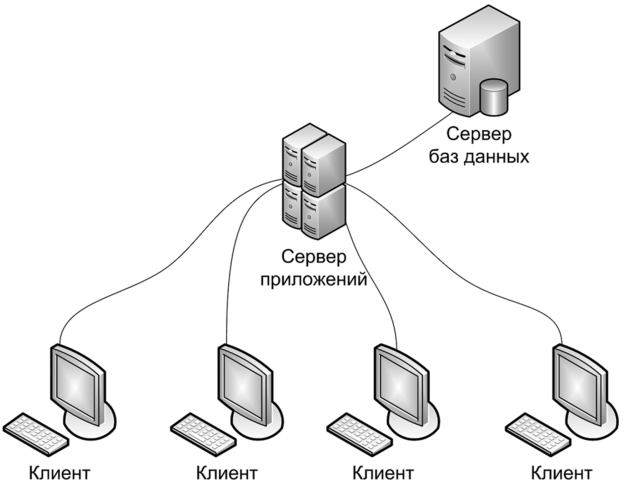
· ORM Dapper - технология программирования, которая связывает [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с концепциями [объектно-ориентированных языков программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), создавая «виртуальную [объектную базу](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) данных»

· Механизм CI/CD - Непрерывная интеграция (Continuous Integration, CI) и непрерывная поставка (Continuous Delivery, CD) представляют собой культуру, набор принципов и практик, которые позволяют разработчикам чаще и надежнее развертывать изменения программного обеспечения

· FIRST - 5 простых принципов которым должны соответствовать юнит тесты. Fast, Independent, Repeatable, Self-Validating, Timely.

**3.3 Разработка программного кода ИС – Предлагаю подробнее описать, что делает код. Чуть ли не каждую строчку**

ASP.Net приложения включают в себя несколько основных частей. При проектировании приложения было принято решение придерживаться стандартной трёхуровневой архитектуры. Это значит, что приложение делится на серверную часть, часть доступа к данным и часть бизнес логики. Пример того, как выглядит работа стандартной реализации трёхуровневого приложения представлен на рисунке…



Рисунок… Трехуровневая архитектура клиент-сервер

Серверная часть в основном состоит из контроллеров. Это класс задачей которого является ожидание и обработка веб-запросов от некоторого клиента. Контроллеры создаются, чтобы предоставить возможность клиенту выполнять базовые(CRUD) операции над некоторой сущностью: чтение, запись, изменение, удаление. Пример контроллера по работе с задачами представлен на рисунке …

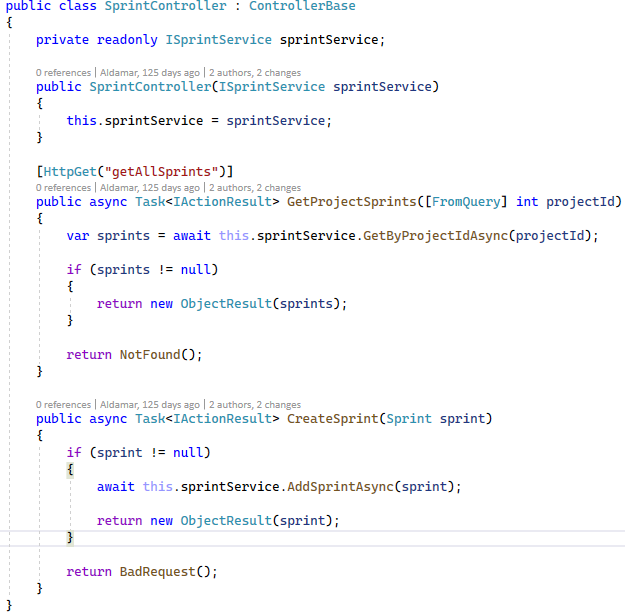


Рисунок … – Фрагмент кода контроллера

Часть бизнес логики представляет собой сервиса, которые выступают прослойкой между серверной частью и частью доступа к данным. Основная роль сервисов, это предварительная обработка данных, полученных из хранилища данных. Обработка происходит перед тем, как отправить данные клиенту. Пример сервиса предоставлен на рисунке…

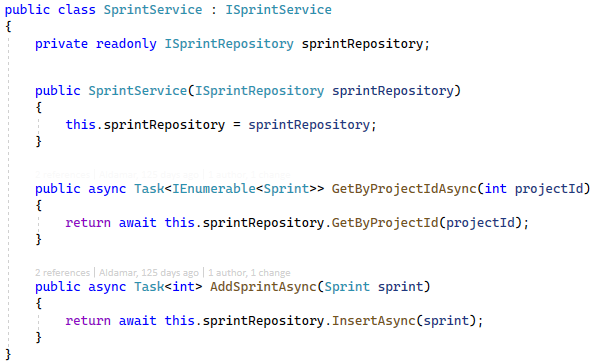
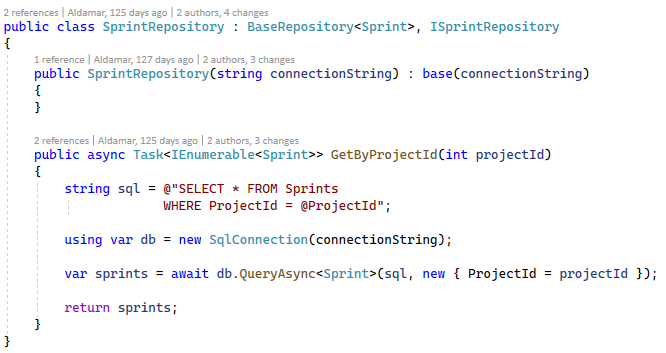


Рисунок 3.3 – Фрагмент кода сервисного класса

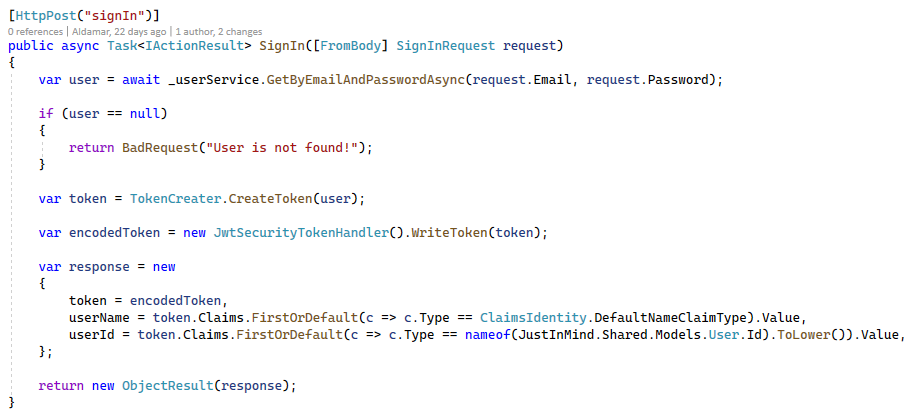
Наконец, на рисунке… представлен пример класса репозитория.



Рисунок… Фрагмент кода класса репозитория

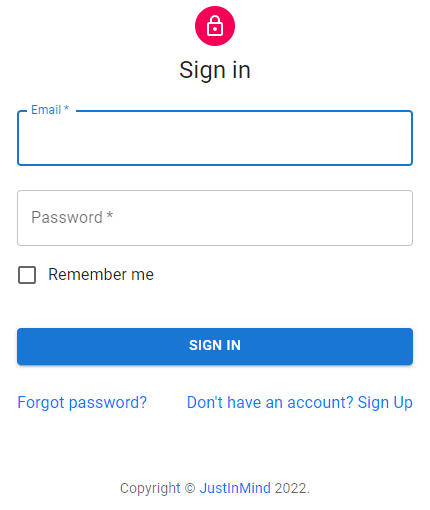
Репозитории осуществляют доступ к хранилищу данных и передают полученные данные на уровень сервисов. В этом слое находится только логика по работе с базой данных и ничего больше. Репозитории должны быть свободны от любой посторонней бизнес-логики. В примере выше описан репозиторий, который получает из базы данных все спринты, относящиеся к определённому проекту.

Отдельного внимания заслуживает и контроллер, отвечающий за авторизацию и аутентификацию пользователя. На рисунке… представлен пример кода, который авторизует пользователя.



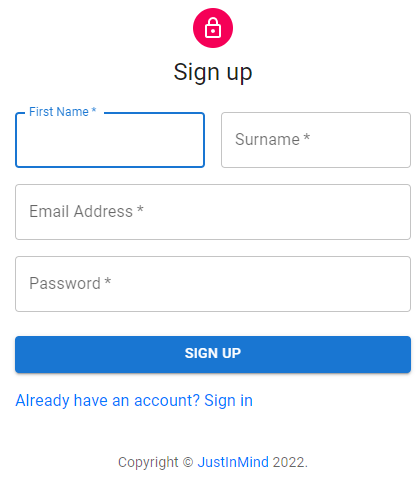
**3.4 Руководство пользователя и контрольные примеры – стоит ли комментировать и показывать каждую кнопку?**

Если приложение запускается локально, то адрес, по которому к нему можно получить доступ может быть установлен системным администратором. Начало работы с приложением происходит через окно входа рисунок…

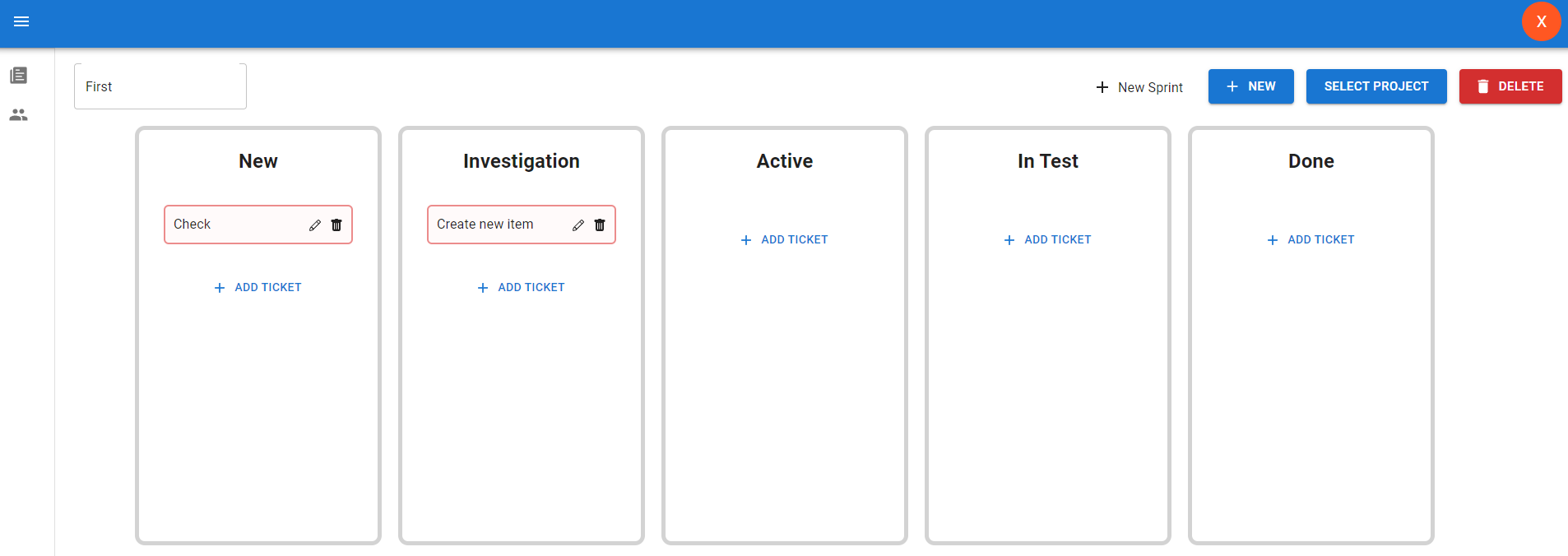


Если у пользователя ещё нет учётное записи, то ему предоставляется возмодность пройти регистрацию, нажав на соответствующую ссылку Sign Up.

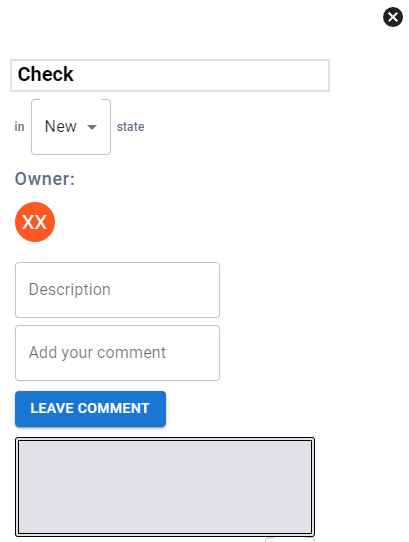
При нажатии на эту ссылку откроется окно регистрации рисунок.



После прохождения процедуры аутентификации и авторизации пользователю представляется основное рабочее окно. Это окно работы с проектом рисунок…

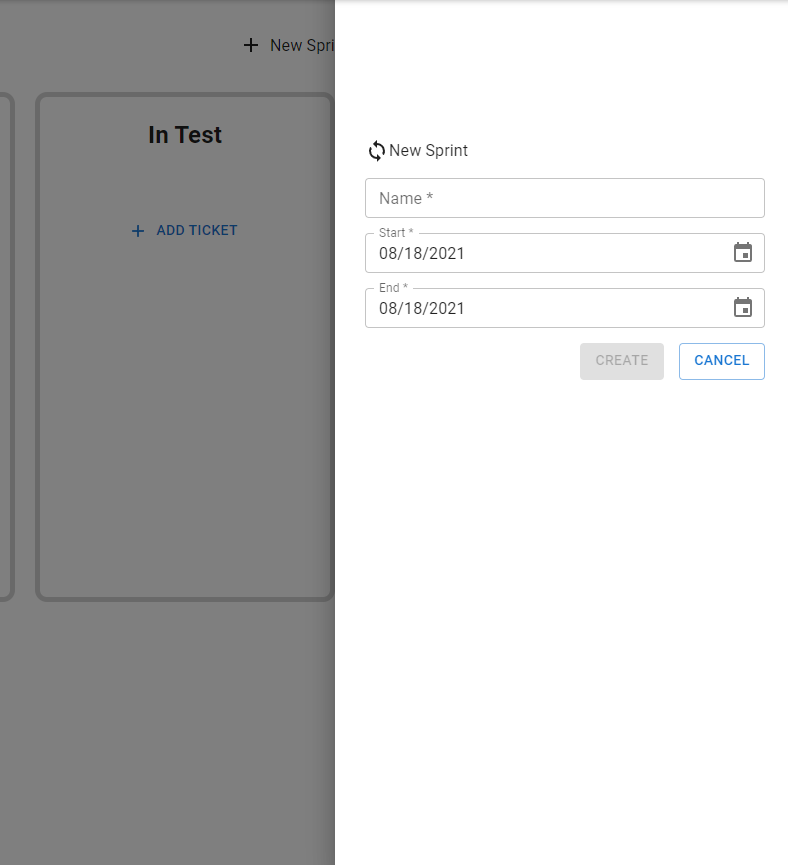


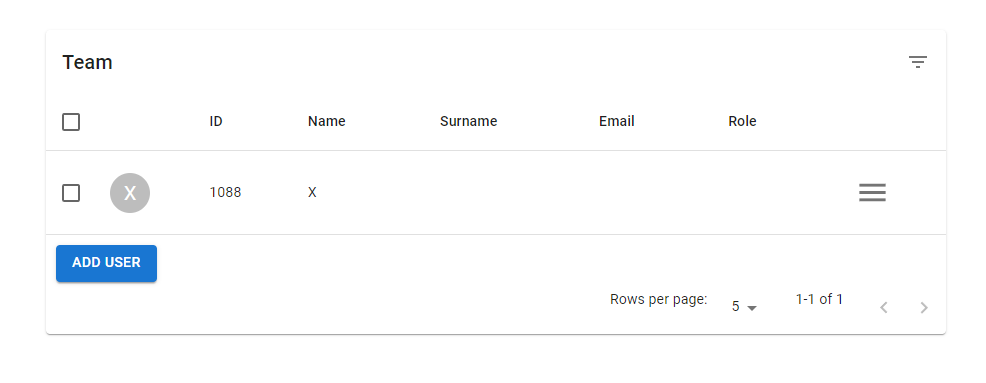
Здесь пользователь может создавать новые проекты либо выбрать уже существующие путём нажатия на соответствующие кнопки. Так же при желании есть возможность удалить проект. На доске задач пользователь может перетаскивать задачи из одного состояния в другое или же при нажатии на рисунок карандаша увидеть полное описание задачи рисунок… .



В этом окне пользователь имеет контроль почти над каждой частью задачи. Есть возможность поменять название, состояние, владельца, описание. Так же можно начать обсуждение, добавив комментарий к задаче.

Если команда жедает вести разработку на основании методолгоий Agile, то у неё есть возможность создавать спринты. Пример всплывающей формы создания спринта рисунок… .



Так же в меню пользователь может увидеть всю команду в виде таблицы рисунок.  


Здесь пользователь может пригласить новых участников, написать сообщение на электронную почту или удалить члена команлы из списка.

**4 Политика информационной безопасности**

**4.1 Цель, принципы и задачи защиты информации в информационной системе**

Информационная безопасность – обеспечение конфиденциальности и целостности информации, недопущение несанкционированных действий с ней, в частности, ее использования, раскрытия, искажения, изменения, исследования и уничтожения.

Главная цель, которая стоит перед системой информационной безопасности является обеспечение защиты данных от внешних и внутренних угроз. Недопущение неправомерного доступа, уничтожения, модификации (изменения), копирования, распространения и (или) предоставления информации, блокирования правомерного доступа к информации, а также иных неправомерных действий

Для обеспечения в информационной системе полной конфиденциальности применяются четыре метода, актуальных для любого формата информации:

* ограничение или полное закрытие доступа к информации;
* шифрование;
* дробление на части и разрозненное хранение;
* скрытие самого факта существования информации.

В данной ИС стоят цели обеспечение правового режима использования информации и информационных ресурсов, обрабатываемых в ИС, как объектов собственности, а также обеспечение и постоянное поддержание в соответствии с условиями, выдвинутыми собственником информационных ресурсов, свойств конфиденциальности, доступности и целостности.

В ИС решена задача снижения риска утечки информации ограниченного доступа в связи с локальным расположением системы и рядом технологий связи, с которыми данная система функционирует.

Решена задача снижения риска несанкционированного воздействия на информацию, система работает с методами защиты данных, разработанными корпорацией Microsoft, они помогают избежать такого рода проблемы в приложении.

**4.2 Методы и средства обеспечения защиты информационных ресурсов**

При разработке данного дипломного проекта не обеспечивалась конфиденциальность используемых данных, т.к. в этом отсутствует острая необходимость, так как используемые данные числу публикаций и мест в рейтинге брались из открытого доступа, также обязанности по обеспечению целостности данных делегируются на системное программное обеспечение, установленное на компьютере пользователя. Примером такой защиты могут выступать учётные записи пользователей, которые требуют ввода пароля при попытке получения доступа, или предустановленная программа BitLocker, которая позволяет уберечь данные в случае утери компьютера.

**4.3 Средства защиты информации и информационных ресурсов**

Для решения защиты информации в разрабатываемой ИС были использованы следующие методы:

* идентификация и аутентификация пользователя;
* шифрование с помощью BitLocker;
* антивирусная защита информационных ресурсов.

**5 Организационно-экономическая часть**

**6 Охрана труда**

**7 Энергосбережение и ресурсосбережение при внедрении новых технологий**

**Заключение**

Список использованных источников

* 1. Маркин, А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин. - М.: Диалог-Мифи, 2014. - 384 c.
  2. [Бхамидипати](http://oz.by/books/more106510.html?id_search=127378#writer), К. SQL. Справочник программиста/ К.Бхамидипати–М.: Эком, 2003. – 304 с.
  3. Бьюли, А. Изучаем SQL / А.Бьюли, Э. Оппель. – М.: Символ, 2007. – 312 с.
  4. Жилинский, А. Самоучитель Microsoft SQL Server 2005 / А.Жилинский – СПб: BHV, 2004. – 224 с.
  5. Павловская, Т. А. С#. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2007. – 432 с.: ил.
  6. Подбельский, В. В. Язык С#. Решение задач / В. В. Подбельский. – М.: Инфра-М, 2014. – 296 с.
  7. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, **2017**. - 480 c.
  8. Еремин И.И., Астафьев Н.Н. Введение в теорию линейного и выпуклого программирования -М.: Наука, 1976 г. – 239 с.
  9. Общий курс высшей математики для экономистов. Учебник / под ред В.И. Ермакова.- М.: ИНФА - М. - 656 с. - (серия «высшее образование»).

10. [https://arman-engineering.ru/baza-znanij-inzhenera-avtomatizirovannaja-sistema-upravlenija-tehnologicheskimi-processami]