Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Современные технологии обработки экономической информации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

БГУИР КП 1-40 05 01 02 010 ПЗ

Студент группы 873603 Портнов Н. В.

Руководитель старший преподаватель кафедры ЭИ

Голда О. А.

Минск 2021

**СОДЕРЖАНИЕ**

Введение 4

1 Использование информационных технологий в сфере управления проектами в IT-компаниях 5

2 Постановка задачи по автоматизированию системы для управления проектами в IT-компаниях 8

2.1 Постановка задачи на разработку системы 8

2.2 Обзор методов решения поставленной задачи 9

3 Функциональное моделирование процесса управления проектами в IT-компаниях 11

4 Построение информационной модели системы для управления проектами в IT-компаниях 15

5 Спецификация вариантов использования системы управления проектами в IT-компаниях 16

6 Модели представления системы управления проектами в IT-компаниях 18

6.1 Диаграмма состояний 18

6.2 Диаграмма последовательностей 18

6.3 Диаграмма компонентов 18

6.4 Диаграмма развёртывания 19

7 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 20

7.1 Схема алгоритма клиент-серверного соединения 20

7.2 Схема алгоритма авторизации 20

8 Руководство пользователя по развёртыванию системы для управления проектами в IT-компаниях 21

9 Результаты тестирования разработанной системы для управления проектами в IT-компаниях 29

Заключение 32

Список использованных источников 33

Приложение А (обязательное) Диаграммы UML 34

Приложение Б (обязательное) Схемы алгоритмов работы программы 36

Приложение В (обязательное) Листинг кода 38

# **ВВЕДЕНИЕ**

Информационные технологии в современном мире интенсивно развиваются и применяются повсеместно. Наиболее широко применяемые средства в наше время – это сотовая связь и интернет, мобильные телефоны и компьютеры. Тем не менее, каждая узкая отрасль и производства имеет своё специфическое оборудование, а также специально разработанное программное обеспечение, обеспечивающее работу устройств.

Цель любого предприятия – получение прибыли и улучшение качества работы для своих сотрудников. Игнорирование, либо нежелание внедрения информационных технологий в текущих реалиях сулят предпринимателю не только уменьшением ее количества, но и делают предприятия неконкурентоспособным по сравнению с теми предприятиями, которые придали данным шагам должное внимание.

Таким образом, была поставлена цель разработать такую систему, которая могла бы позволить минимизировать финансовые затраты за счет автоматизации процесса в сфере управления проектами it компаний.

Разрабатываемая система должна обеспечить возможность создавать проекты и задачи, назначать на них сотрудников команды, которая будет разрабатывать данный проект.

Для достижения поставленной цели необходимо решить следующие задачи:

* осуществить анализ использования информационных технологий в сфере проектного менеджмента (реализации своих услуг посредством интернета);
* создать базу данных для хранения различных проектов;
* реализовать серверную часть, которая позволит работать с базой данных;
* разработать для пользователя простой и удобный интерфейс приложения;
* протестировать разработанное приложение;
* разработать и описать руководство пользователя.

# **1 ИСПОЛЬЗОВАНИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В СФЕРЕ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

Как и в других современных отраслях, ключевым фактором успеха в ИТ-сфере является грамотное управление проектами, что под силу только высококвалифицированному руководителю, обладающему необходимыми компетенциями в данной области.

С ростом количества it компаний также увеличилось количество проектов. Поскольку одной из важнейших целей любой компании является прибыль, то при разработке нужно распределить обязанности разработчиков, которые будут непосредственно заниматься проектами компании как личными, так и под заказ.

Чтобы разработчикам было удобно распределить свое время, данная система должна упростить создание задач на проекте, а также внесение изменений по эстимейту задачи и количеству человек, которое необходимо для ее решения.

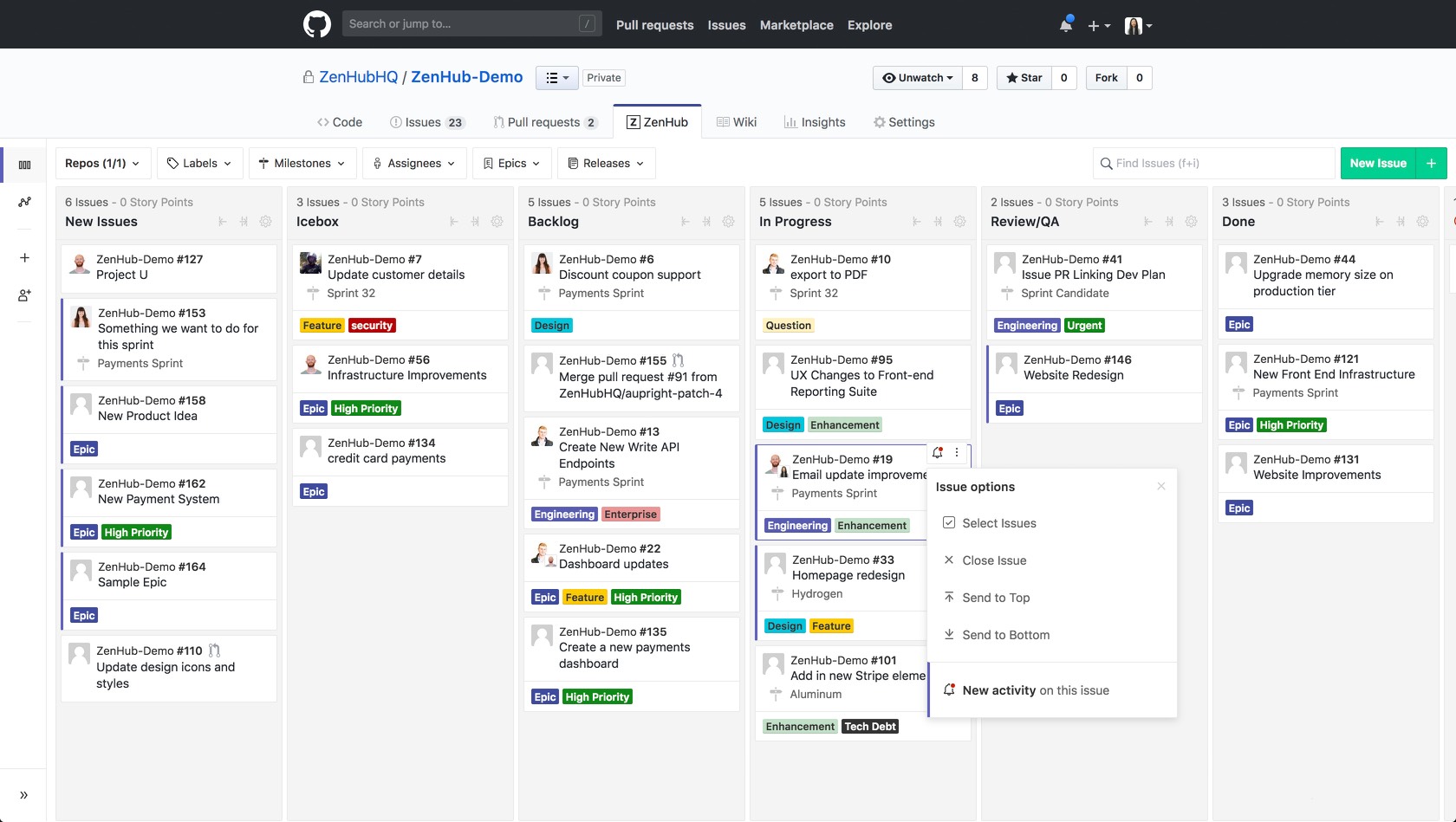


Рисунок 1.1 – Пример реализации системы управления проектами на примере ZenHub

[Управлять людьми](https://worksection.com/systema-upravlenyja-personalom-v-kompanyy.html) сложно. Управлять людьми эффективно — неимоверно сложно. Поэтому мы изобретаем и смешиваем разные практики, алгоритмы и технологии контроля, оценки результатов труда. Переходим от устаревших методик к новым, стараемся учесть все нюансы и риски, тратим время и ресурсы на планирование вместо работы по накатанной.

Из множества методологий нужна одна оптимальная и настроенная лично под нас и наш проект. Одно дело выполнять шаблонные задачи строго по скриптам из книги, другое — методом проб и ошибок вырабатывать свои специальные инструменты для учета приоритетов компании. Поэтому проектный менеджмент (PM), то есть методология управления компанией с делением всей работы на проекты, становится популярным во всех отраслях.

Многие компании только сейчас переходят от классической (отработанная, часто бюрократическая схема) к проектной (каждая задача отдельно, делегирование ответственности) модели управления. Общий менеджмент для предпринимателя сводился к тому, что есть руководитель и исполнитель задачи. Дисциплина взаимодействия зависела от того, как прописано в шаблоне. Шаблон же был вбит в разумы всех одинаково лет этак 20-30 назад. Только этот шаблон уже не применить к новым условиями мирового рынка.

Исторически все началось с того, что в период перестройки 90-х годов сама логика ведения бизнеса была нарушена настолько, что попытки построить новую рабочую концепцию управления в миллениум создали термин «компания-однодневка». Концепции таких компаний проверялись на прочность и только одна из 10 компаний существовала больше полугода в 2000-х. Параллельно оставались многолетние предприятия, кардинально меняясь каждые лет 5 чтобы выжить в период перемен.

В 2010-х информационный взрыв интернета сделал доступной всю разрозненную базу наработок европейского и американского бизнеса. Из тонн полезной и мусорной информации вроде «что такое управление проектами и современный менеджмент», «как распределять обязанности, правильно предугадывать и сокращать риски» предприниматели до сих пор выбирают крупицы знаний, применимых именно в их компаниях.

Сейчас мы следим в новостях как прибыльные корпорации растут и распадаются на подразделения, покупаются и переживают структурные слияния. За каждым актом купли-продажи отдела кроется сложная сеть проектов (гаджеты, приложения, расширения для браузеров), связанных по разным критериям. Когда меняются критерии, меняется деление проектов на группы. Именно изменение группировки и сообщают нам в СМИ. Внутри же компании продолжают работать, а проекты ведутся непрерывно.

Главный выбор предпринимателя при разработке программ и приложений — это подходящая методология управления. Их действительно много, на 2017-й год существуют:

* [Waterfall](https://worksection.com/blog/waterfall.html).
* [Управление критической цепи проекта (CCPM)](https://worksection.com/blog/ccpm.html). Фокус на ресурсах, не сроках.
* [Метод критического пути (CPM)](https://worksection.com/blog/it-project-management.html). «Шаг за шагом» алгоритмы.

PMI / PMBOK «Метод». Инициирование, планирование, исполнение, контроль и закрытие. Инструкция, не метод по сути.

Гибкая методология:

* [Agile](https://worksection.com/blog/agile.html).
* [Scrum](https://worksection.com/blog/scrum.html).
* [Kanban](https://worksection.com/blog/kanban.html).
* [Экстремальное программирование (XP)](https://worksection.com/blog/extreme-programming.html).
* [Адаптивные рамки проекта (APF)](https://worksection.com/blog/adaptive-project-framework.html).

Методики по управлению изменениями:

* [Методология моделирования событий (ECM)](https://worksection.com/blog/ecm.html).
* [Экстремальное управление проектами (XPM)](https://worksection.com/blog/extreme-project-management.html).

Разрабатывать и внедрять PM в компании стоит постепенно, проверяя на практике каждый этап и взаимодействие.

За один день перейти на новые стандарты — нереально. Даже чтобы внедрить новый органайзер нужно: обучить команду пользоваться, перенести дела и задачи в такс-менеджер, назначить ответственных и дедлайны, настроить баг-трекеры. Обычно коллективу тяжело привыкнуть к изменениям, да и интеграция баз данных занимает время. Хотя в Worksection уже давно есть функционал для переноса данных из других сервисов, а гайд для новичков и сама система настолько интуитивно понятны, что это отмечают наши постоянные клиенты в отзывах.

Все процессы управления проектами также происходят постепенно и проходят 5 этапов:

* Разработка концепции, инициирование.
* Определение и планирование.
* Запуск работы и воплощение задуманного.
* Контроль и наблюдение.
* Закрытие проекта.

# **2 ПОСТАНОВКА ЗАДАЧИ ПО АВТОМАТИЗИРОВАНИЮ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

**2.1 Постановка задачи на разработку системы**

В данном курсовом проекте поставлена цель повышение качества по управлению проектами, разбив его на задачи. Для этого необходимо разработать такое приложение, которое предоставит возможность многим пользователям добавлять новые имея уже существующие задачи и проекты. В перспективе, компания, оказывающая услуги в сфере проектного менеджмента, сможет привлечь большое количество пользователей на свою платформу, и тем самым сможет повысить свою прибыль и популярность среди it компаний. В этом заключается коммерческий смысл разработки данного проекта.

Предлагается разработать приложение в архитектуре клиент-сервер. Так как необходимость воспользоваться приложением будет возникать сразу у нескольких пользователей, сервер должен поддерживать возможность одновременно обрабатывать запросы этих пользователей. Серверное приложение должно производить обработку следующих запросов:

* авторизация пользователя;
* регистрация пользователя;
* получение списка проектов;
* получение списка задач для проекта;
* получение списка пользователей, работающих над задачей;
* получение списка стадий, на которых находятся задачи;
* изменение дат и сроков задач
* создание собственных проектов/задач;
* назначение новых пользователей на задачу/проект;
* сохранение изменений;

Для максимального удобства клиентское приложение должно быть реализовано в виде веб-сайта. Предполагается реализация функционала для пользователей и администраторов. Администратору будут доступны функции просмотра и удалений пользователей, проектов и задач. Простой пользователь в лице слушателя курсов сможет работать с проектами, создавать/удалять/изменять задачи, назначать/удалять пользователей с задач текущего проекта.

**2.2 Обзор методов решения поставленной задачи**

Для программной реализации системы в данном курсовом проекте был выбран мультипарадигменный язык программирования JavaScript.

Основные преимущества JavaScript как языка программирования:

* ни один современный браузер не обходится без поддержки JavaScript;
* с использованием написанных на JavaScript плагинов и скриптов справится даже не специалист;
* использует модель DOM, которая обеспечивает множество предустановленных функций для различных объектов на страницах;
* безопасность;
* простота;
* интерпретируемость;
* высокая производительность.

Ещё одним неоспоримым преимуществом JavaScript является его распространённость. Этот язык можно встретить буквально повсюду. Он поддерживается на всех операционных системах, во всех видах браузеров, и на настольных компьютерах, и на мобильных устройствах. Очень важно также и то, что JavaScript-приложения работают без установки их на компьютеры пользователей.

Графический интерфейс клиентской части реализуется с помощью библиотеки React.js. Он представляет современный прогрессивный фреймворк, написанный на языке JavaScript и предназначенный для создания веб-приложений клиентского уровня. Основная сфера применения данного фреймворка — это создание и организация пользовательского интерфейса.

Одним из ключевых моментов в работе React.js является виртуальный DOM. Структура веб-страницы, как правило, описывается с помощью DOM (Document Object Model), которая представляет организацию HTML-элементов на странице. Для взаимодействия с DOM применяется JavaScript. Но когда мы пытаемся манипулировать элементами с его помощью, то можем столкнуться со снижением производительности, особенно при изменении большого количества элементов. А операции над элементами могут занять некоторое время, что неизбежно скажется на пользовательском опыте.

Для этого React.js использует виртуальный DOM. Виртуальный DOM представляет легковесную копию обычного DOM. Если приложению нужно узнать информацию о состоянии элементов, то происходит обращение к виртуальному DOM. Если данные, которые используются в приложении React.js, изменяются, то изменения вначале вносятся в виртуальный DOM. Потом React выбирает минимальный набор компонентов, для которых надо выполнить изменения на веб-странице, чтобы реальный DOM соответствовал виртуальному. Благодаря виртуальному DOM повышается производительность приложения.

Соединение между серверной и клиентскими частями должно устанавливаться с помощью протокола TCP/IP. Этот протокол обладает одним важным преимуществом: он обеспечивает аппаратную независимость. Так как в сетевых протоколах определяется только блок передачи и способ его отправки, TCP/IP не зависит от особенностей сетевого аппаратного обеспечения, позволяя организовать обмен информацией между сетями с различной технологией передачи данных. Система IP-адресов, в свою очередь, позволяет без затруднений установить соединение между любыми двумя машинами сети. Взаимодействие между устройствами в рамках стека TCP/IP осуществляется с помощью связки IP-адреса и номера порта.

Эта связка образует сокет – программный интерфейс, который обеспечивает обмен данными между устройствами на низком уровне. Протокол TCP/IP основывается на соединениях, устанавливаемых между двумя компьютерами, обычно называемых клиентом и сервером. Поэтому различают клиентский сокет и серверный сокет. Для организации взаимодействия клиент должен знать IP-адрес и номер порта сервера, по которым он осуществляет подключение к удаленному устройству.

Для хранения информации используется база данных PostgresQL.  
Это свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Описанные технологии и программные средства характеризуются в первую очередь надёжностью и простотой использования. Совместное их применение позволит создать удобное многофункциональное приложение, отвечающее всем поставленным требованиям.

# **3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССА УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

Проведённый анализ предметной области даёт возможность разработать функциональную модель процесса управления проектами в it компании на основе методологии IDEF0.

На рисунке 3.1 представлена контекстная диаграмма верхнего уровня. Как можно заметить, на входе расположены пользователь и заказ на создание проекта. Необходимыми для успешного выполнения процесса ресурсами являются администрация компании, руководитель проекта и программное средство. Управление процессом осуществляется Внутренним регламентом компании и правилом продукта. Результатом выполнения процесса являются проект с задачами и прибыль компании.

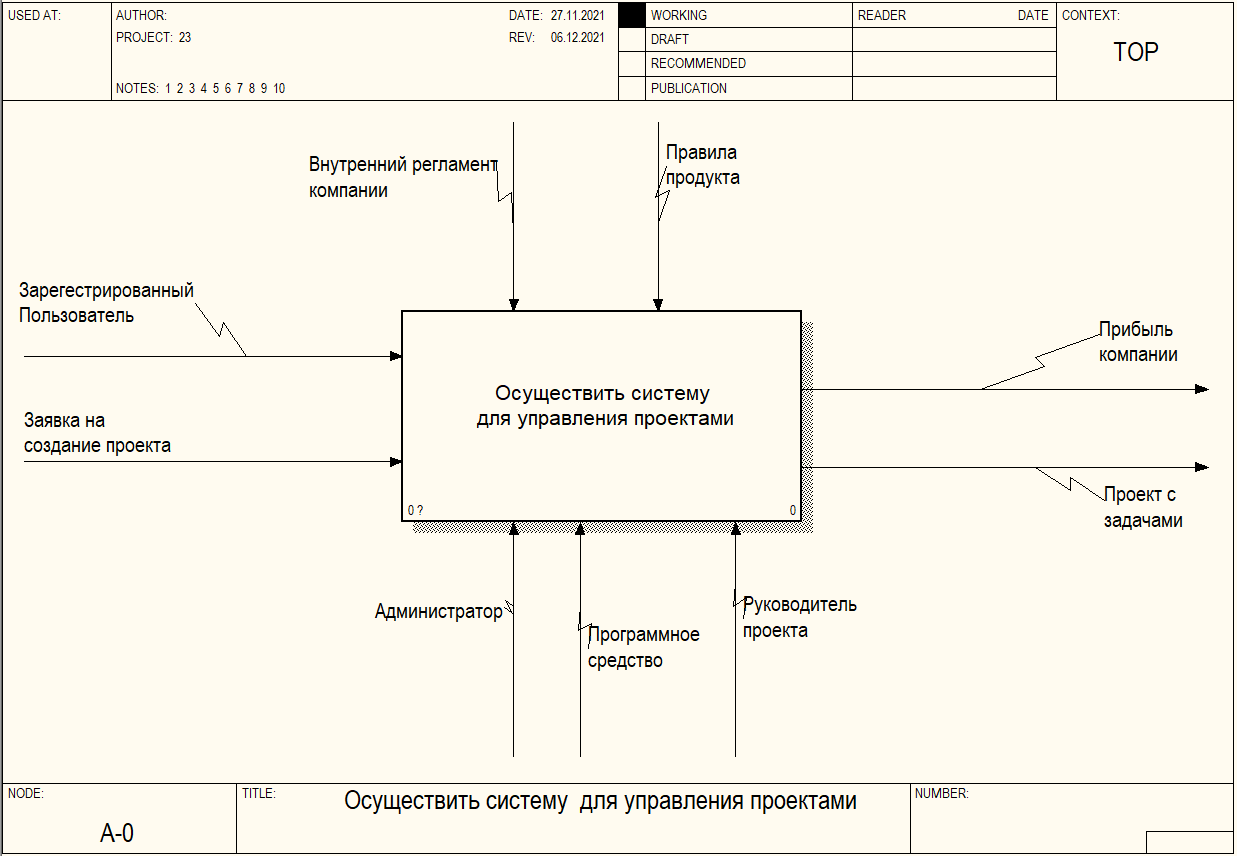


Рисунок 3.1 – Контекстная диаграмма верхнего уровня

После декомпозиции контекстной диаграммы получаем следующие блоки, показанные на рисунке 3.2:

* Пройти регистрацию;
* Создать проект;
* Создать задачу;
* Приступить к выполнению;

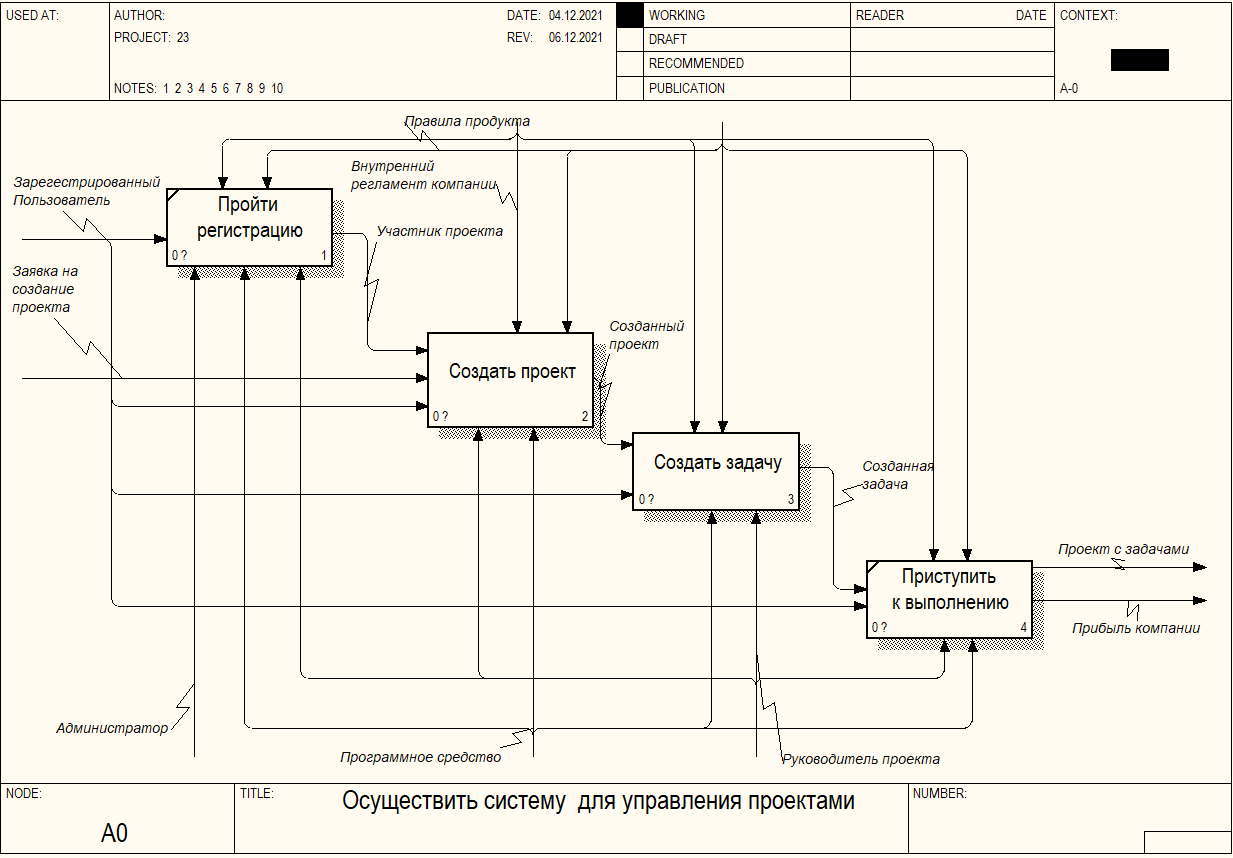


Рисунок 3.2 – Декомпозиция контекстной диаграммы верхнего уровня

Когда администрация компании получает заявку на создание проекта, то ему предоставляется возможность выбрать курс для изучения. После этого он обязан пройти регистрацию. Результатом выполнения блока «Пройти регистрацию» является зарегистрированный пользователь. Последний поступает на вход в блок «Создать проект». В случае успешного создания проекта пользователь может так же создать задачу и приступить к ее выполнению. В результате выполнения всех описанных действий пользователь получает готовый проект, а компания получает прибыль, что и является главной целью осуществления коммерческой деятельности.

Декомпозиция блока «Освоить программу курса» содержит три блока:

* + Ввести название задачи, оценку по времени и описание;
  + Назначить исполнителей;
  + Сохранить данные.

Данная декомпозиция отражена на рисунке 3.3.

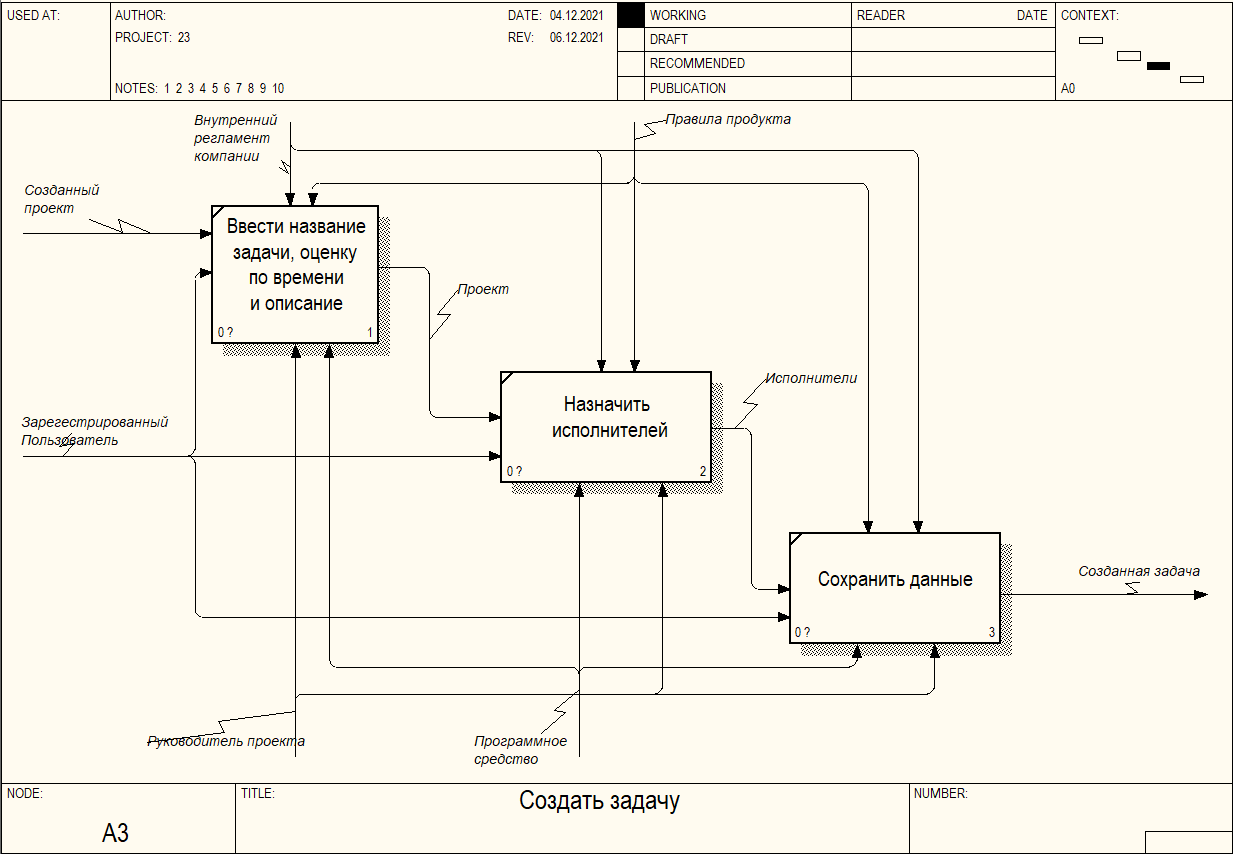


Рисунок 3.3 – Декомпозиция блока «Создать задачу»

Начинается этот процесс с ввода обязательных данных касаемо задачи. Затем выбираются пользователи, которые были так же добавлены в проект и будут выполнять поставленную задачу. После ввода всей необходимой информации нужно сохранить данную задачу в базу данных.

Рассмотрим декомпозицию блока «Создать проект» (рисунок 3.4). В результате её проведения получаются четыре блока:

* Войти в мои проекты;
* Нажать кнопку создать;
* Ввести данные;
* Зайти в проект.

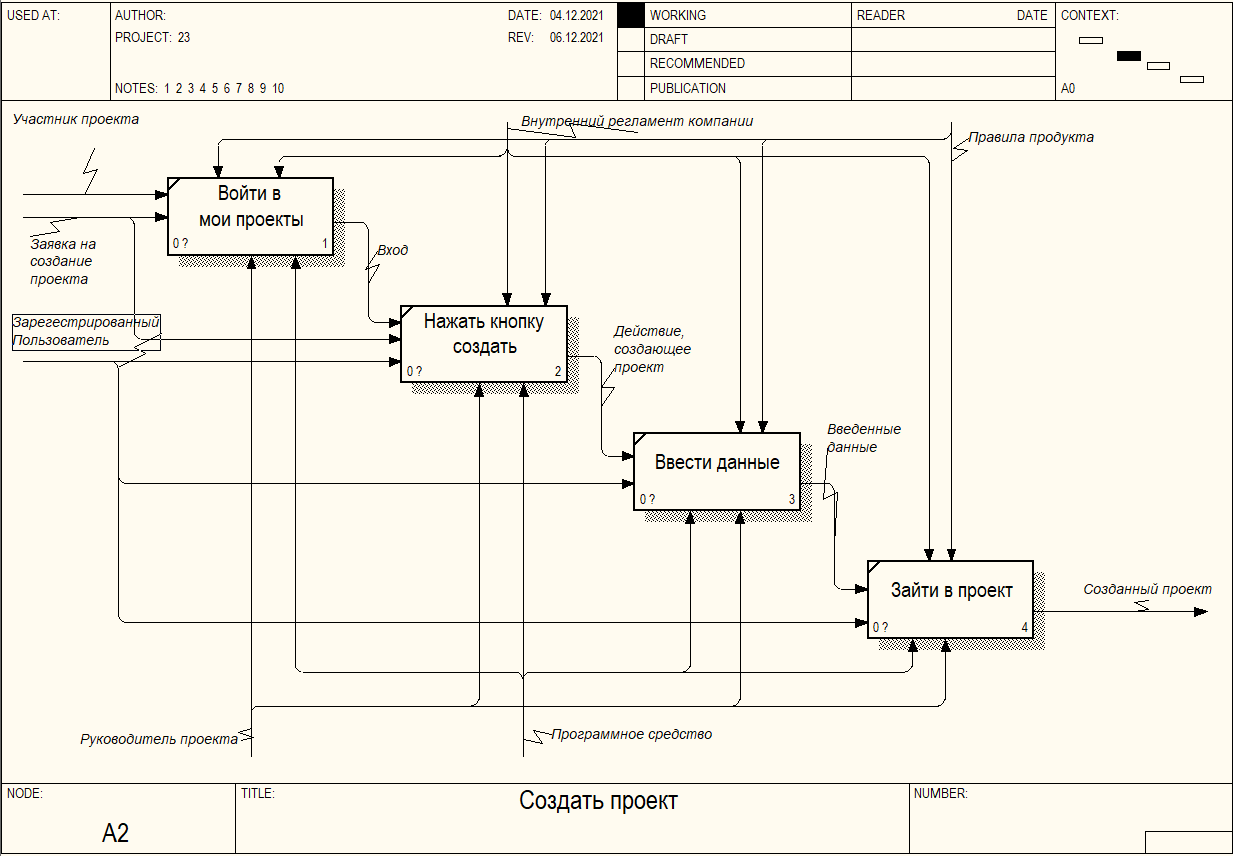


Рисунок 3.4 – Декомпозиция блока «Создать проект»

Был рассмотрен и наглядно продемонстрирован процесс разработки проекта. Хоть процесс и является довольно насыщенным, но с использованием разработанного программного средства он становится намного менее трудоёмким.

# **4 ПОСТРОЕНИЕ ИНФОРМАЦИОННОЙ МОДЕЛИ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

В процессе информационного моделирования были выделены следующие сущности:

* Пользователь;
* Роль;
* Проект;
* Задача;
* Пометка;
* Роль пользователя;
* Задача пользователя;
* Категория;

Сущность «Пользователь» служит для создания аккаунта пользователя. Наличие такой сущности позволяет предоставить ему возможность полноценно создавать задачи и проекты с использованием системы. Атрибутами этой сущности являются почта, пароль и уникальный ID для каждого пользователя.

Для разделения ролей используется сущность Роль. Она разделяет функционал между администратором и пользователем. Атрибутами этой сущности являются значение, описание и уникальный ID для каждой роли.

Сущность «Проект» является важнейшей, поскольку она содержит основные данные, под которые данный курсовой проект и разрабатывался. Атрибутами являются название, описание, уникальный ID, а также данная таблица связана с таблицами Задачи, Пользователи и Пометки.

Сущность «Задача» служит для создания задач на проекте. Атрибутами являются название, описание, оценка задачи по времени, категория, уникальный ID и ID проекта.

Сущность «Пометка» служит для создания пометок на задачах. Атрибутами являются значение, цвет и уникальный ID и ID задачи.

Сущность «Категория» служит для создания категорий задач на проекте. Атрибутами являются значение и уникальный ID.

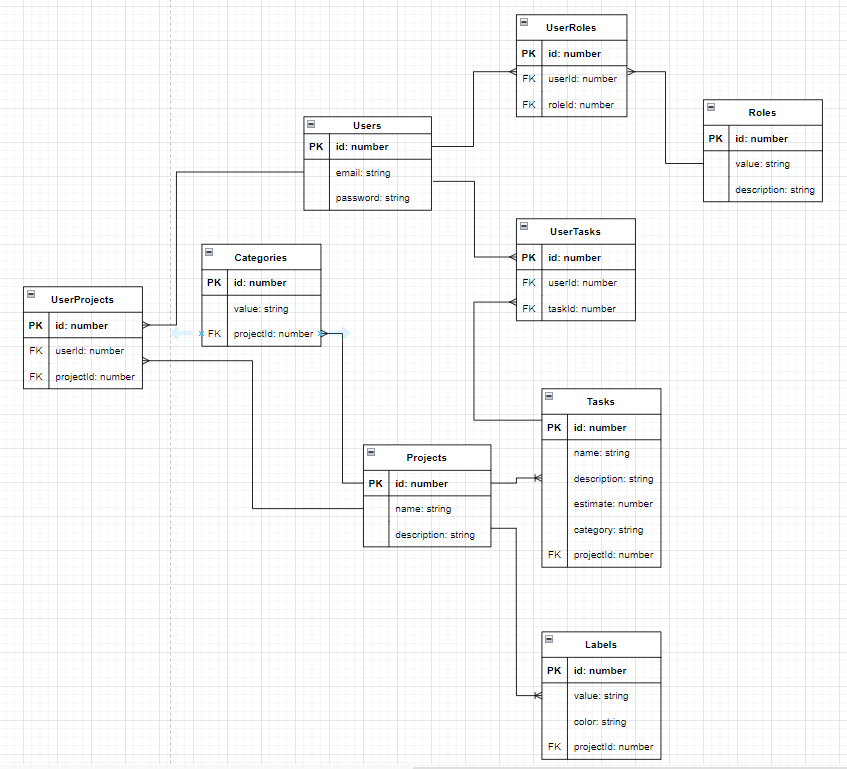


Рисунок 4.1 – Информационная модель базы данных

С учётом обозначенного взаимодействия сущностей смоделируем их взаимодействие в формате IDEF1.X и приведём эту модель к третьей нормальной форме. В результате последовательного приведения получается модель, соответствующая условиям третьей нормальной формы – не ключевой атрибут сущности функционально зависит только от всего первичного ключа и ни от чего другого (рисунок 4.1).

# **5 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования служит для описания сервисов, которые система предоставляет актеру. Диаграмма вариантов использования разрабатываемой системы представлена на рисунке 5.1.

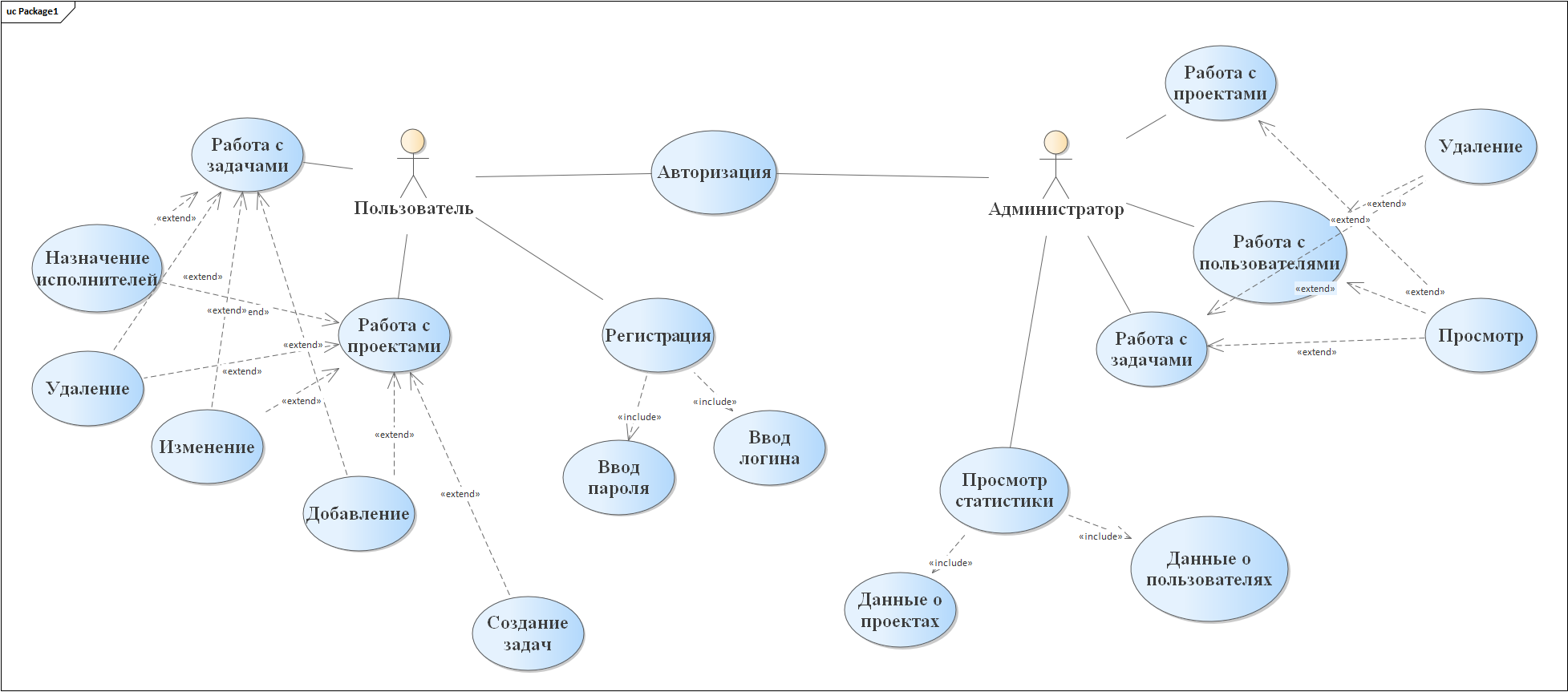


Рисунок 5.1 – Диаграмма вариантов использования

В данном проекте были выделены 2 актёра – пользователь и администратор.

Администратор обладает пятью вариантами использования – авторизация в системе, просмотр статистики, а также работа с задачами, проектами и пользователями. Работа из последних перечисленных вариантов использования включает в себя две составляющие – это просмотр и удаление.

Учащемуся доступны четыре вариантов использования. Среди них выделяется работа с проектами и задачами, авторизация и регистрация. Также учащийся может назначать новые задачи другим пользователям, просматривать, изменять, удалять и добавлять новые задачи или же проекты.

# **6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИИ**

**6.1 Диаграмма состояний**

Диаграмма состояний является широко известным средством описания поведения систем. Она определяет все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Данная диаграмма полезна при моделировании жизненного цикла объекта. Диаграмма состояний разрабатываемой системы представлена на рисунке А.1.

На начальном этапе работы системы происходит создание задачи. Далее

Выбираются исполнители данной задачи и записывается оценка задачи по времени. После выполнения задачи ее отправляют в завершенные и ждут результата от тестировщика. Если задача выполнена с точки зрения бизнес-логики, то считается выполненной.

**6.2 Диаграмма последовательностей**

Эта диаграмма является видом диаграмм взаимодействия, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования. Основываясь на диаграмме, разработанной и описанной в главе 5, была построена диаграмма последовательностей, которая показана на рисунке А.2.

Начинается работа с ввода пользователем логина и пароля. Эти данные передаются на сервер, и сервер выполняет запрос к базе данных, тем самым проверяя наличие такого пользователя. Если в базе данных такой пользователь находится, то происходит авторизация, серверу возвращается результат этой авторизации.

На следующем этапе пользователь создает проекты, информация отправляется на сервер, выполняется запрос на добавление, и запись о проекте с соответствующими данными добавляется в базу данных. База данных отправляет серверу результат операции добавления.

**6.3 Диаграмма компонентов**

Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами. В качестве таких компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и так далее. Рассмотрим диаграмму компонентов для системы работы веб-приложения по управлению проектами в IT компании, которая изображена на рисунке А.3.

Модуль domain (у сайта может измениться доменное имя, поэтому используется данное понятие) отвечает за функционирование клиентской части приложения. Этот модуль представлен в виду веб-страницы и пользовательский интерфейсом.

Модуль main.ts реализует работу серверной части приложения. Компоненты этого модуля отвечают за установку клиент-серверного соединения, обработку запросов клиента и работу с модулем базы данных. Для взаимодействия с базой данных необходим сервер. Приложение устанавливает связь с базой данных PostgresQL посредством Node Nest.js.

**6.4 Диаграмма развёртывания**

Диаграмма развертывания показывает топологию системы и распределение компонентов системы по её узлам, а также соединения – маршруты передачи информации между аппаратными узлами. На рисунке А.4 показана диаграмма развёртывания разрабатываемой системы.

Как можно заметить, основными элементами являются любое устройство клиента с доступом в Internet и сервер приложения. Устройством может являться и ПК, и смартфон, и планшет, так как верстка сайта является адаптивном.

Сервер включает в себя исполняемый файл и систему управления базами данных PostgresQL. Взаимодействие между клиентскими компьютерами и серверным узлом осуществляется посредством стека протоколов TCP/IP.

# **7 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ**

**7.1 Схема алгоритма клиент-серверного соединения**

В данном курсовом проекте реализована архитектура «клиент-сервер». К преимуществам данной архитектуры, как правило, относят высокую защищённость данных, централизацию доступа к хранимым данным, быстродействие, быстроту обслуживания и обработку данных. Схема алгоритма клиент-серверного-соединения представлена на рисунке Б.1.

Программное средство разработано таким образом, что графический интерфейс реализуется на клиентской части, которая формирует и отправляет запросы серверу. Сервер же, в свою очередь, формирует запросы к базе данных. После выполнения запросов результаты возвращаются на клиентскую часть.

Работа программы начинается с установления соединения с сервером. Затем происходит проверка корректности установленного соединения. Далее, при условии успешного подключения к серверу, происходит переход к авторизации. В случае, если пользователь не имеет учётной записи в системе, то ему необходимо создать её. Это делается с помощью функции регистрации. Информация, введённая пользователем, сохраняется в базу данных. После создания учётной записи пользователь может произвести вход в систему. Сделав это, у него появляется возможность совершить какое-то действие, например, создать проект и задачи для него. Для этого необходимо нажать на кнопку создать проект и ввести необходимые данные. После чего пользователь может завершить работу с системой.

**7.2 Схема алгоритма создания задачи**

Схема алгоритма данного процесса представлена на рисунке Б.2. Выполнение этого алгоритма начинается с ввода названия, описания и времени затрат на задачу в форму для отправки. Сервер получает это значение и создаёт запрос к базе данных. Затем происходит установка соединения с базой данных. После этого сервер отправляет созданный запрос и ждёт получения ответа. Ответ приходит в виде выборки задач текущего проекта с учетом добавленной. В случае ошибки будет выведено соответствующее сообщение.

# **8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО РАЗВЁРТЫВАНИЮ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИЯХ**

Для проверки работоспособности приложения необходимо совершить следующие действия:

* сделать git clone Github-репозитория;
* в терминале прописать cd server && npm install && npm run start:dev, затем в новом терминале cd client && npm i && npm start;
* перейти в браузер и в адресной строке ввести localhost:3000.

После перехода на веб-страницу пользователь увидит форму авторизации, которая изображена на рисунке 8.1. Для входа в систему необходимо ввести свои логин и пароль.

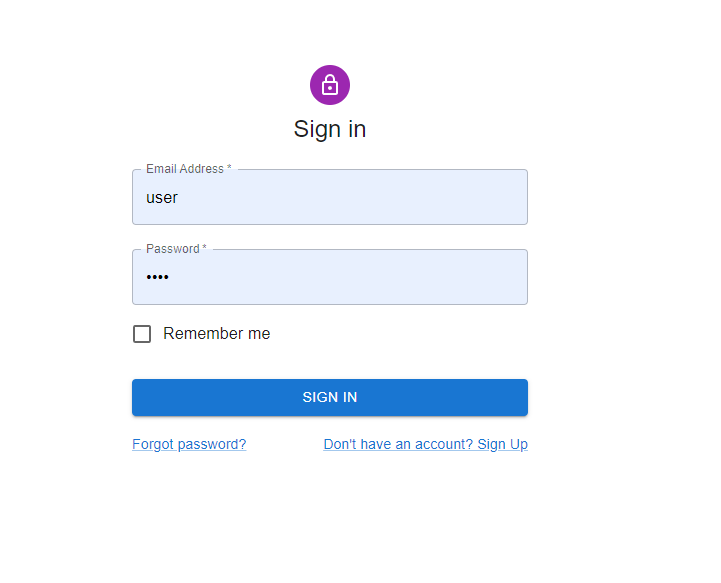


Рисунок 8.1 – Авторизация пользователя

Если пользователь не имеет учётной записи в системе, то ему необходимо зарегистрироваться. Это делается путём нажатия на вкладку «Регистрация» и дальнейшего ввода своих данных в форме, представленной на рисунке 8.2.

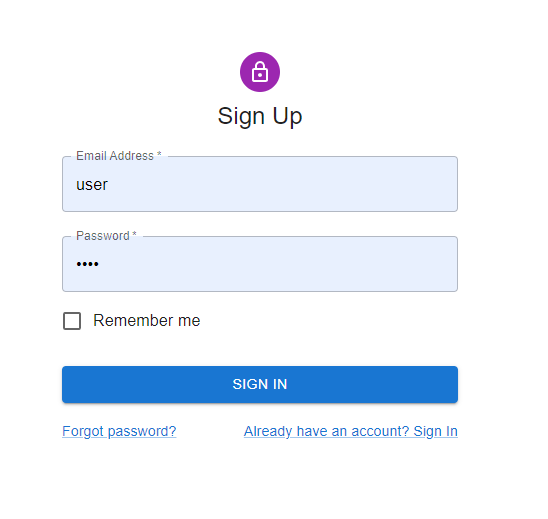


Рисунок 8.2 – Регистрация пользователя

При успешном входе в систему перед пользователем будет открыта главная страница приложения, показанная на рисунке 8.3.

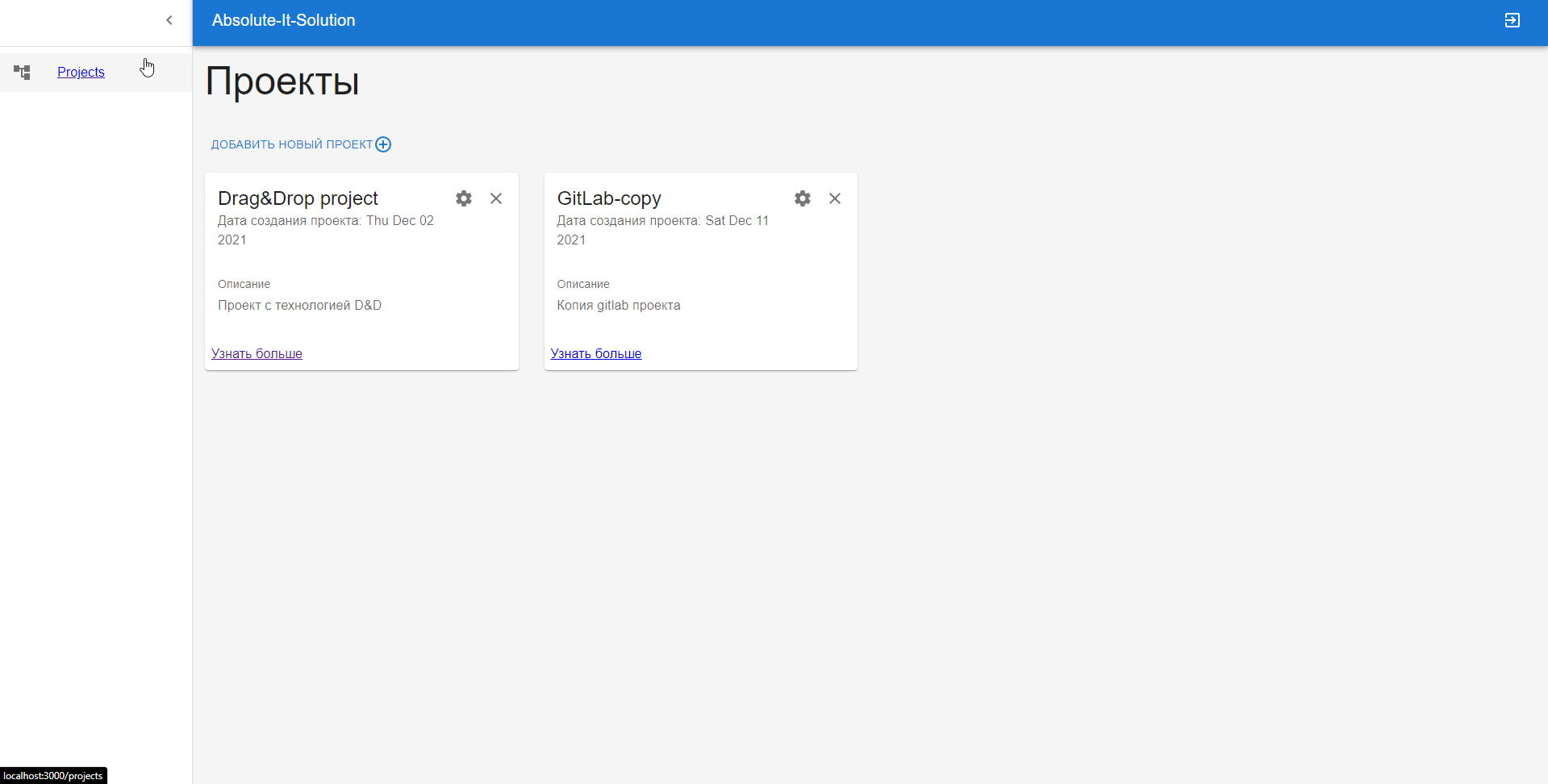


Рисунок 8.3 – Страница с проектами

На главной странице представлен список проектов, доступных пользователю для разработки. Также в левой части главной страницы есть меню вкладок. По нажатию на кнопку «Узнать больше» выбранного курса происходит переход на страницу этого проекта. Общий вид такой страницы изображён на рисунке 8.4.

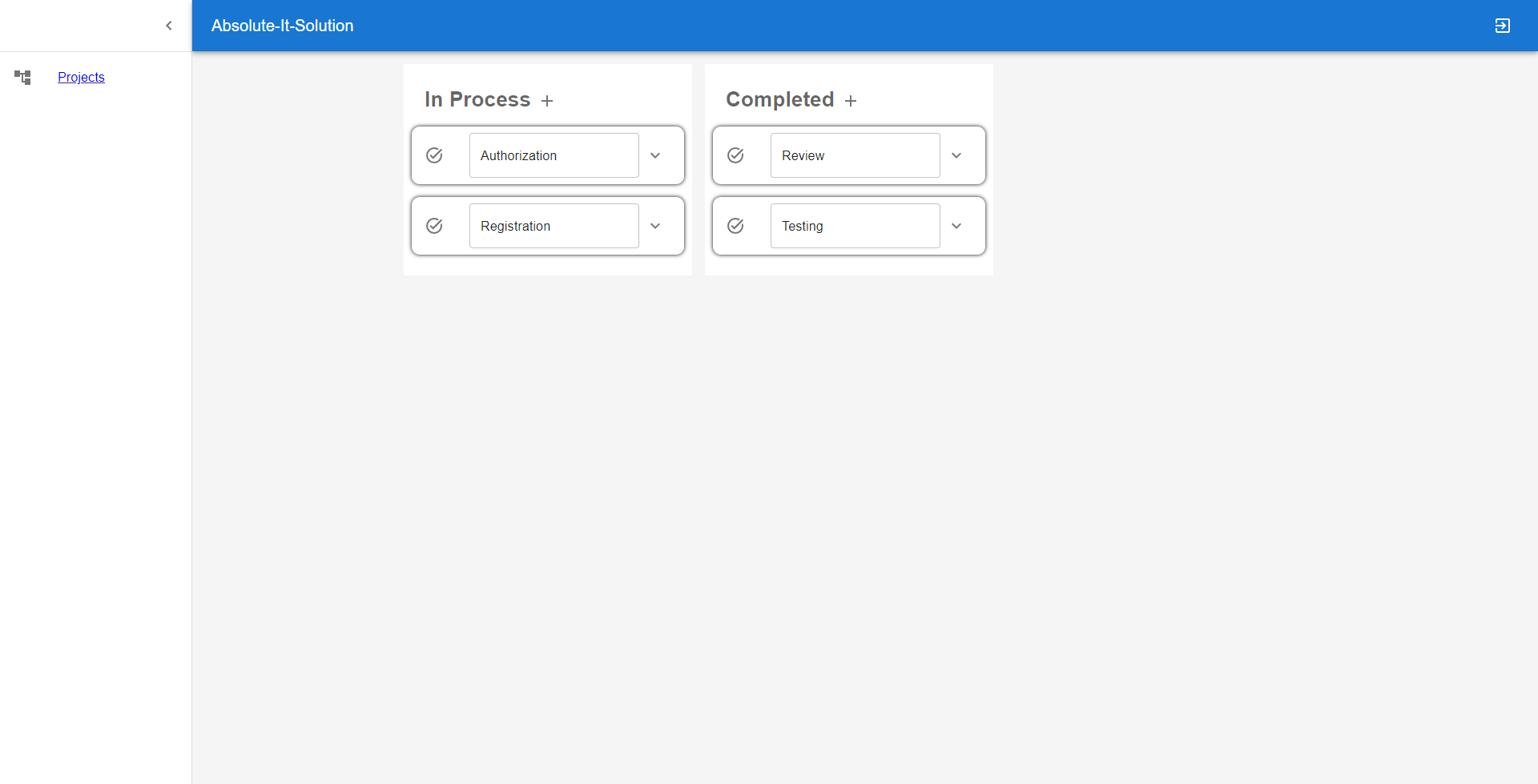


Рисунок 8.4 – Страница проекта

Данная страница имеет несколько категорий, которые имеют задачи. Так же можно добавить новые в существующие категории соответствующего проекта, пример которой представлен на рисунке 8.5.

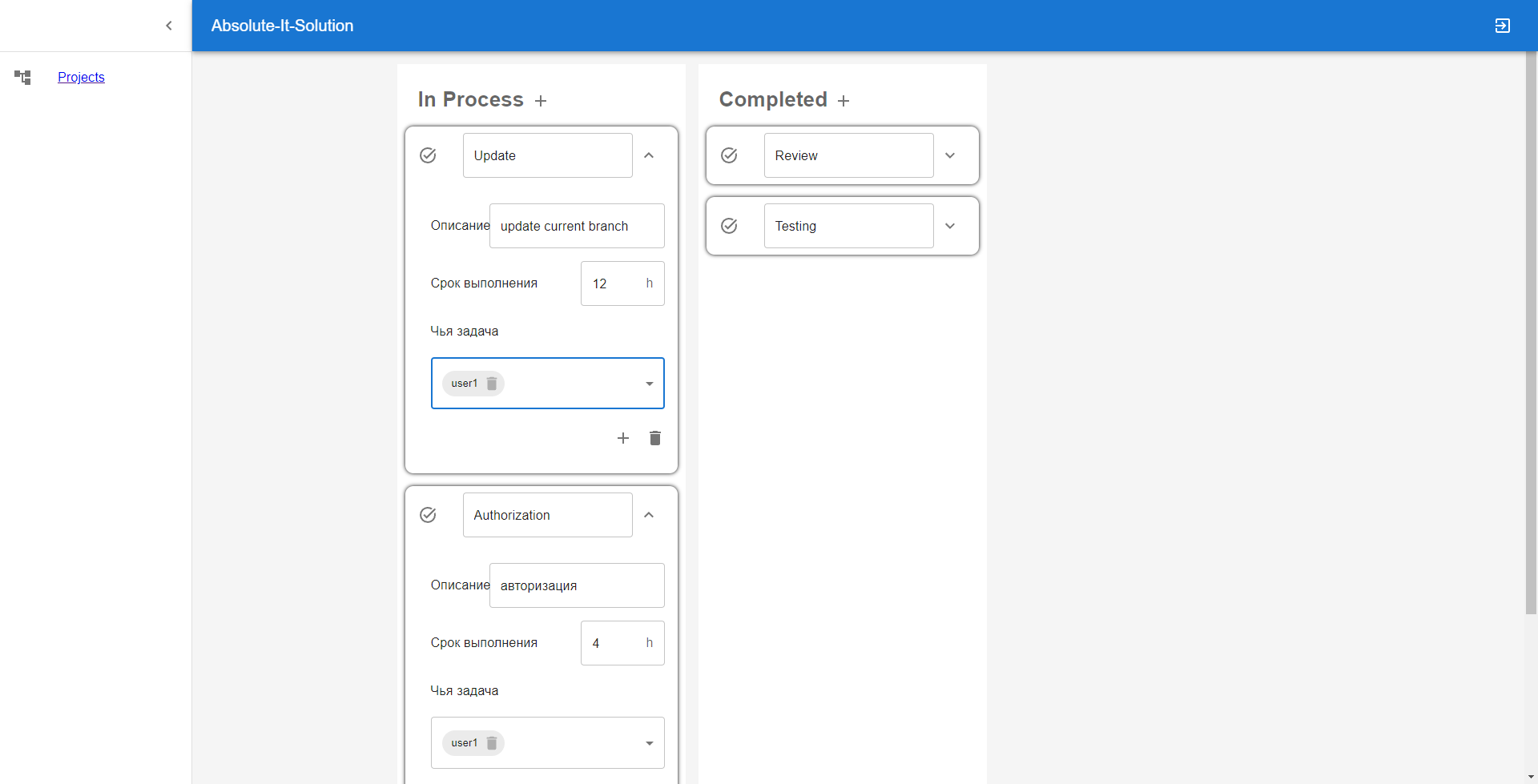


Рисунок 8.5 – Добавление новой задачи

После добавления, данная задача будет внесена в текущий проект. Данный процесс показан на рисунке 8.6.

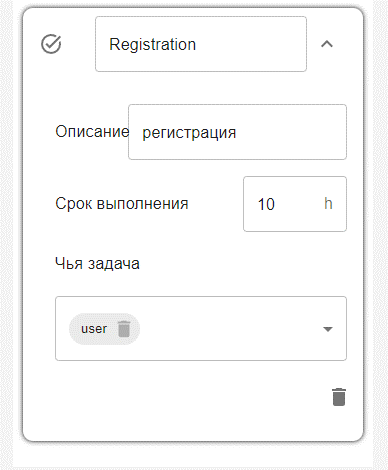


Рисунок 8.6 – Созданная задача

По нажатию на кнопку «Удалить задачу» пользователь ее удалит, результат изображён на рисунке 8.7.

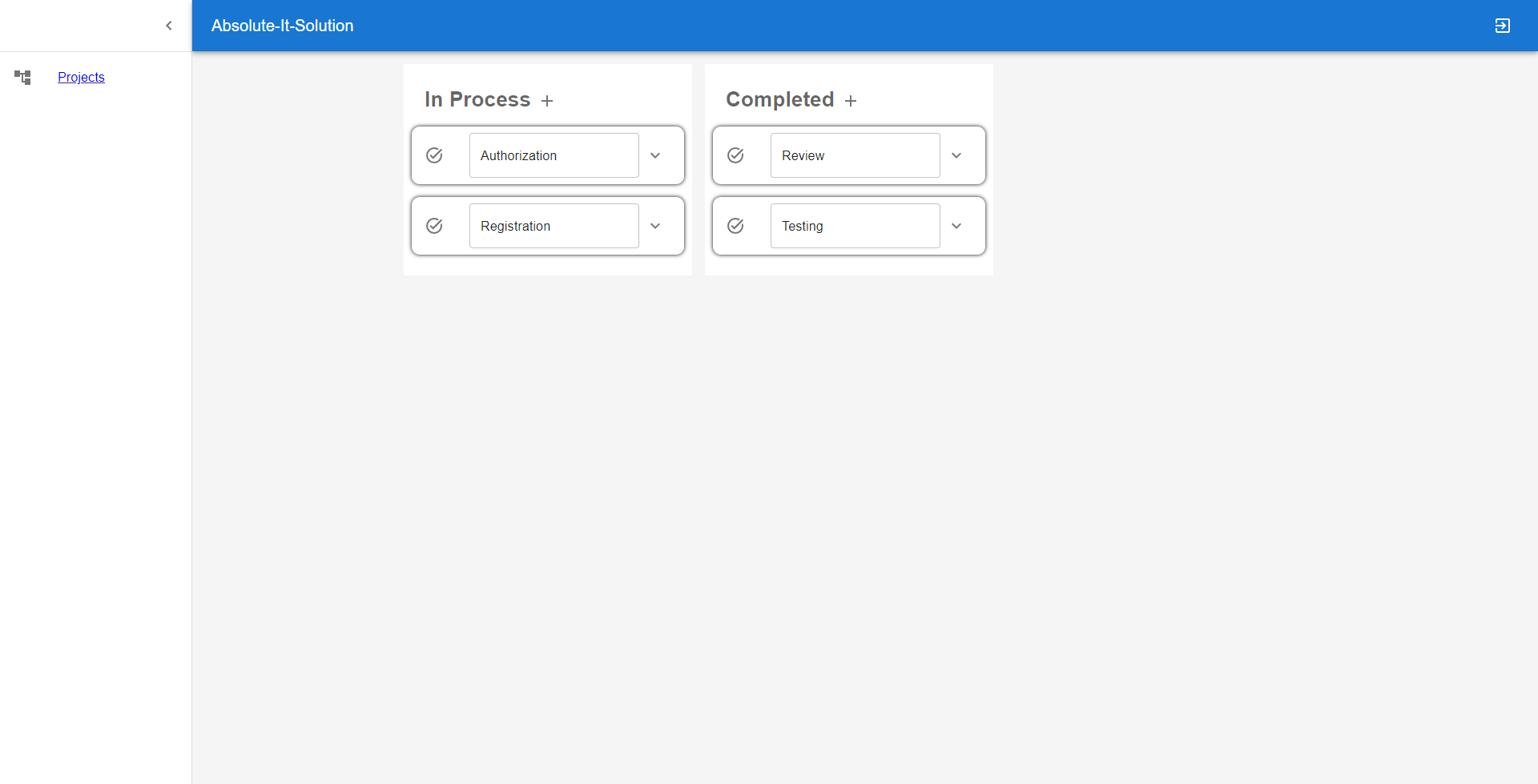


Рисунок 8.7 – Список задач без удаленной

Если пользователь захочет внести изменения в существующую задачу, то нужно нажать на кнопку сохранить задачу(с иконкой дискеты) на рисунке 8.8.

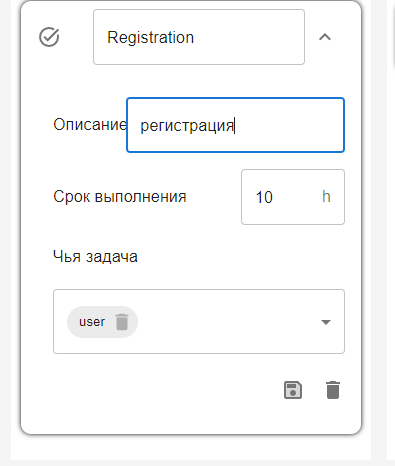


Рисунок 8.8 – изменение текущей задачи

После нажатия на кнопку «Сохранить изменения» текущая задача будет изменена, результат изображён на рисунке 8.9.

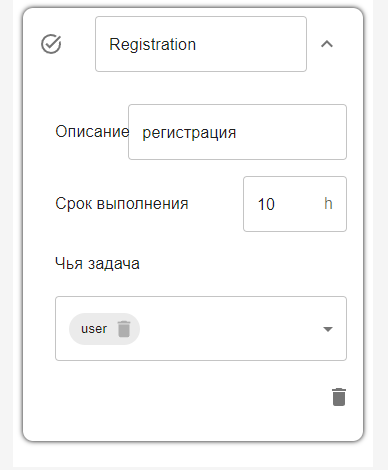


Рисунок 8.9 – Измененная задача

нажать на кнопку «Отправить». После отправки это решение попадает в список проверенных, который можно увидеть во вкладке «Proven Solutions».

# **9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ В IT-КОМПАНИЯХ**

В процессе работы данного приложения по разным причинам могут возникать различные ошибки. Для устойчивого функционирования необходимо предусмотреть обработку исключительных ситуаций.

Как уже упоминалось в предыдущем разделе, для входа в систему пользователю необходимо пройти авторизацию. В случае, если был введён логин, на который не зарегистрирована учётная запись, появится соответствующее уведомление. Данное уведомление представлено на рисунке 9.1.

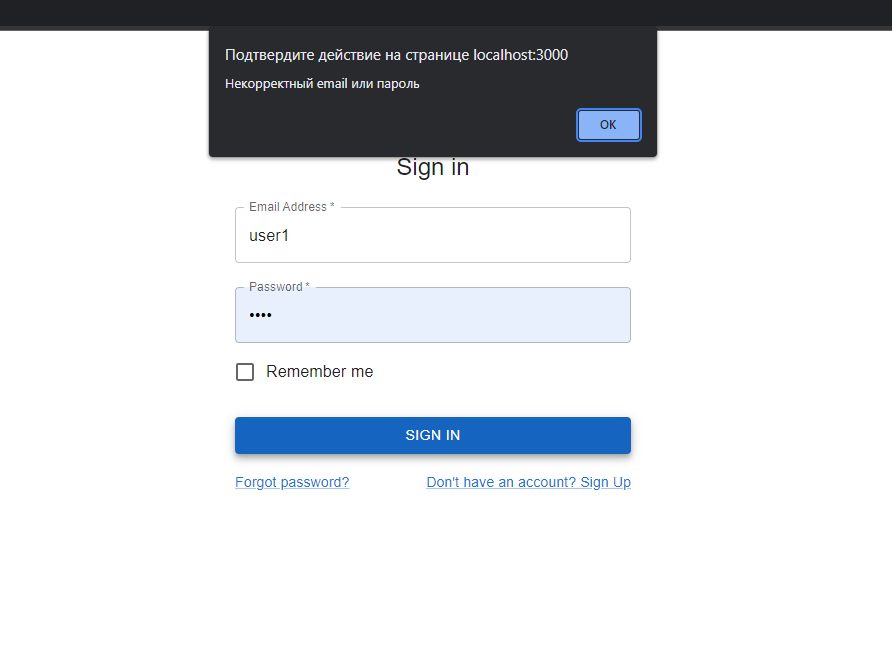


Рисунок 9.1 – Сообщение о вводе неверного логина или пароля

Если пользователь хочет зарегистрироваться и введет почту уже хранящуюся в базе данных, то увидит соответствующее сообщение (рисунок 9.2).

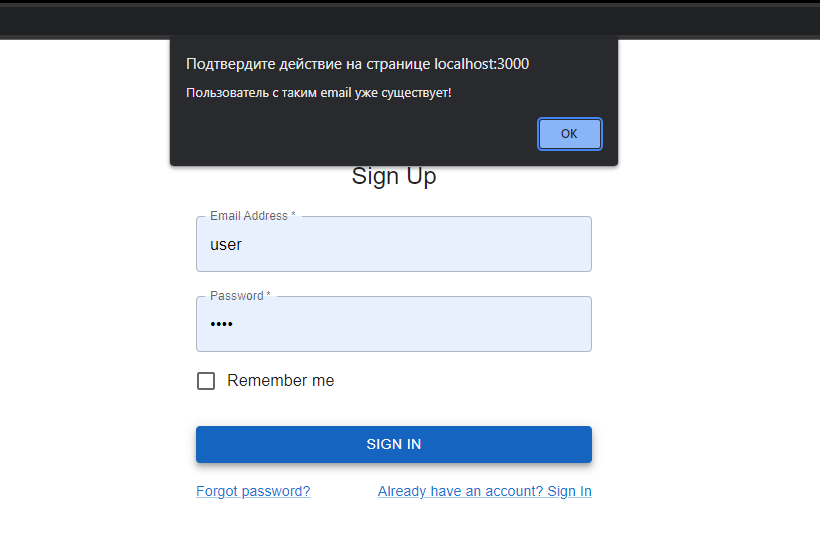


Рисунок 9.2 – Сообщение об уже существующем пользователе

Если пользователь хочет создать новый проект с существующим названием, то получит ошибку (рисунок 9.3).

# **ЗАКЛЮЧЕНИЕ**

В результате проделанной работы было создано веб-приложение, дающее возможность пользователю очень простым и удобным способом организовать разработку проектов для it-компаний. Все это возможно делать в удалённом режиме, что экономит время пользователя.

В ходе создания программного средства были подробно изучены особенности внедрения информационных технологий в сферу проектного менеджмента. Полученная информация позволила построить функциональную модель IDEF0, которая наглядно отображает процесс управления проектами с помощью веб-приложения. Были показаны и описаны диаграммы UML, с помощью которых было выполнено проектирование системы. Также была рассмотрена архитектура созданного программного средства. Помимо этого, в ходе выполнения проекта было составлено руководство пользователю, где понятным и доступным языком описывается принцип работы. В завершение работы было проведено тестирование разработанной системы, подтвердившее работоспособность созданного программного средства.

Основные функции программного средства реализованы в соответствии с выявленными особенностями предметной области. Был разработан довольно широкий функционал для работы с информацией, которая содержится в базе данных. Стиль интерфейса программы создавался с упором на массовость потребления и использования, который позволит любому пользователю легко и удобно разобраться содержимым на сайте.

Главными из отличий данного программного средства является надёжная и безопасная база данных, простой и понятный интерфейс для взаимодействия.

В будущем возможно рассмотрение вопроса о расширении функционала программы или же усовершенствования имеющегося. Это обеспечит расширение спектра применения разработанного программного средства.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Академия профессионального развития [Электронный ресурс]. – Электронные данные. – Режим доступа: https://academy-prof.ru/blog/informacionnye-tehnologii-v-medicine.

[2] Учебник Javascript [Электронный ресурс]. – Электронные данные. – Режим доступа: https://learn.javascript.ru.

[3] Habr [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/post/152477/>

[4] Бюро социальной информации [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://ru.belbsi.by/rights/council/theses/?tid=2837>.

[5] MDN web docs [Электронный ресурс]. – Электронные данные. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript.

[6] w3schools [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.w3schools.com/jquery/>.

[7] Nest.js [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://nestjs.com/>

[8] Мир JavaScript 2021 [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/company/ruvds/blog/542376/>.

[9] Выборка из БД [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/nosql/mongodb/2.4.php>.

[10] Веб-фреймворк Express [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs>.

[11] Руководство по PostgresQL [Электронный ресурс]. – Электронные данные. Режим доступа: <https://www.postgresql.org/>.

[12] Руководство по React [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.reactjs.org/>.

[13] Руководство по Redux [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://tproger.ru/translations/redux-for-beginners/>.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Диаграммы UML**

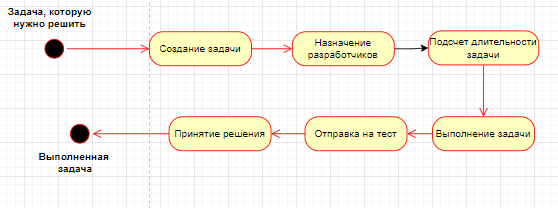


Рисунок А.1 – Диаграмма состояний

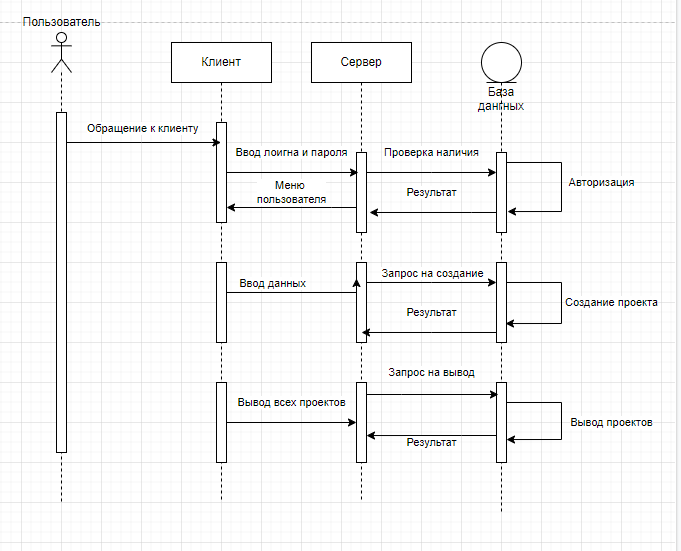


Рисунок А.2 – Диаграмма последовательностей

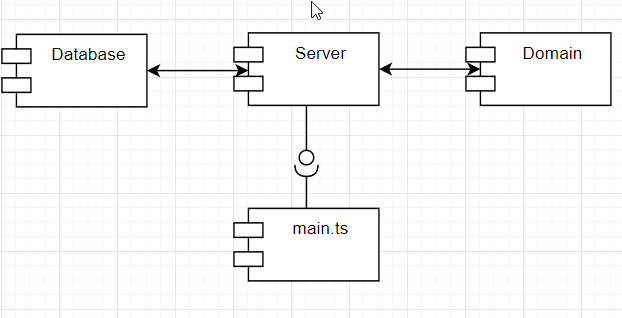


Рисунок А.3 – Диаграмма компонентов

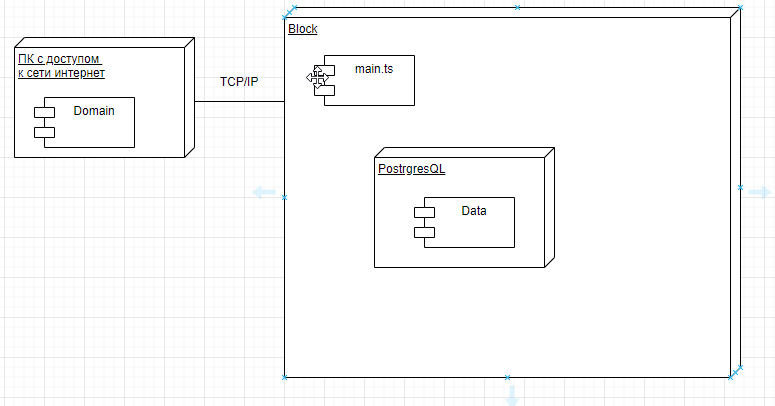


Рисунок А.4 – Диаграмма развёртывания

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Схемы алгоритмов работы программы**



Рисунок Б.1 – Схема алгоритма клиент-серверного взаимодействия



Рисунок Б.2 – Схема алгоритма создания задачи

# **ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Листинг кода**

import { NestFactory } from '@nestjs/core'

import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger'

import { AppModule } from './app.module'

import { JwtAuthGuard } from './auth/jwt-auth.guard'

async function start() {

    const PORT = process.env.PORT || 5000

    const app = await NestFactory.create(AppModule)

    const config = new DocumentBuilder()

        .setTitle(`stoei backend api`)

        .setDescription(`api, backend, stoei`)

        .setVersion('0.0.1')

        .addTag(`Portnov N.V. 873603 backend`)

        .build()

    const document = SwaggerModule.createDocument(app, config)

    SwaggerModule.setup('/api/docs', app, document)

    // app.useGlobalGuards(JwtAuthGuard)

    app.enableCors({

        origin: true,

        allowedHeaders:

            'X-Requested-With, X-HTTP-Method-Override, Content-Type, Accept, Observe',

        methods: 'GET,PUT,POST,DELETE,UPDATE,OPTIONS',

        credentials: true,

    })

    await app.listen(PORT, () => {

        console.log(`server started on port ${PORT}`)

    })

}

start()