МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И   
ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ**

**Кафедра информационных технологий и электронного обучения**

Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения – очная

**Курсовая работа**

«Разработка программного проекта сервиса для управления задачами»

Обучающегося 3 курса

Моисеенко Павла Александрович

Научный руководитель:

Кандидат физико-математических наук,   
доцент кафедры ИТиЭО

Жуков Николай Николаевич

Санкт-Петербург

2020

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc61400955)

[ГЛАВА I. ПРОЦЕСС РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 4](#_Toc61400956)

[1.1. Определение понятия 4](#_Toc61400957)

[1.2. Различные парадигмы и модели 4](#_Toc61400958)

[1.3. Инструменты и фреймворки гибкой методологии разработки 7](#_Toc61400959)

[ВЫВОД К ГЛАВЕ I 9](#_Toc61400960)

[ГЛАВА II. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МЕТОДОЛОГИИ 10](#_Toc61400961)

[2.1. Техническое задание 10](#_Toc61400962)

[2.2. Диаграмма Ганта 11](#_Toc61400963)

[2.3. Макет 12](#_Toc61400964)

[2.4. Инструмент управления проектом 14](#_Toc61400965)

[2.5. UML-диаграмма 15](#_Toc61400966)

[2.6. Тестирование 17](#_Toc61400967)

[ВЫВОД К ГЛАВЕ II 18](#_Toc61400968)

[ЗАКЛЮЧЕНИЕ 19](#_Toc61400969)

[ЛИТЕРАТУРА 20](#_Toc61400970)

# ВВЕДЕНИЕ

Разработка программного обеспечения — это трудо- и времязатратный процесс. Чтобы это сделать максимально эффективно, стоит внедрить один из методов управления разработкой программного обеспечения. Сейчас появляется всё больше разработчиков, которые разрабатывают ПО и которым нужны для этого подходящие им методы разработки. Поэтому мной были описаны различные модели и фреймворки разработки программного обеспечения, для каких случаев они подходят и пример использования на основе моего проекта разработки программного проекта сервиса для управления задачами.

# ГЛАВА I. ПРОЦЕСС РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 1.1. Определение понятия

В программной инженерии процесс разработки программного обеспечения — это процесс разделения работы по разработке программного обеспечения на отдельные фазы для улучшения проектирования, управления продуктом и управления проектами. Он также известен как жизненный цикл разработки программного обеспечения (SDLC). Методология может включать предварительное определение конкретных результатов и артефактов, которые создаются и завершаются командой проекта для разработки или сопровождения приложения. Большинство современных процессов разработки можно скорее назвать гибкими. Но есть и другие методологии такие как водопад, прототипирование, итеративная и инкрементная разработка, спиральная разработка, быстрая разработка приложений и экстремальное программирование. «Модель» жизненного цикла иногда считается более общим термином для категории методологий, а «процесс» разработки программного обеспечения — более конкретным термином, относящимся к конкретному процессу, выбранному конкретной организацией.

## 1.2. Различные парадигмы и модели

Для эффективного управления проектом менеджер или группа разработчиков должны выбрать методологию разработки программного обеспечения, которая лучше всего подходит для данного проекта. Все методики имеют разные сильные и слабые стороны и существуют по разным причинам. Ниже описаны наиболее часто используемые методологии разработки программного обеспечения и их плюсы и минусы.

Команды используют методологию гибкой разработки (Aglie), чтобы минимизировать риски (например, ошибки, перерасход средств и изменение требований) при добавлении новых функций. Во всех гибких методах команды разрабатывают программное обеспечение итерациями, которые содержат мини-приращения новой функциональности. Существует множество различных форм метода гибкой разработки, включая Scrum, Канбан, экстремальное программирование (XP) и разработка, управляемая функциями (FDD).

Основное преимущество гибкой разработки программного обеспечения состоит в том, что она позволяет выпускать программное обеспечение итерациями. Итерационные выпуски повышают эффективность, позволяя командам находить и исправлять дефекты и согласовывать ожидания на ранней стадии. Они также позволяют пользователям получать программное обеспечение с частыми постепенными улучшениями. Из минусов можно отметить то, что методы гибкой разработки полагаются на общение в режиме реального времени, поэтому новым пользователям часто не хватает документации, необходимой для быстрого освоения. Они требуют от пользователей огромных затрат времени, потому что разработчики должны полностью выполнять каждую функцию в каждой итерации для утверждения пользователем.[1]

Методы гибкой разработки похожи на быструю разработку приложений и могут быть неэффективными в крупных организациях. Программисты, менеджеры и организации, привыкшие к каскадному методу, могут столкнуться с трудностями при адаптации к Agile. Поэтому им часто подходит гибридный подход.[2]

DevOps — это не только методология разработки, но и набор практик, поддерживающих организационную культуру. Развертывание DevOps сосредоточено на организационных изменениях, которые расширяют сотрудничество между отделами, отвечающими за различные сегменты жизненного цикла разработки, такие как разработка, обеспечение качества и операции. Плюсы: DevOps ориентирован на сокращение времени выхода на рынок, снижение количества отказов новых выпусков, сокращение времени между исправлениями и минимизацию сбоев при максимальной надежности. Для этого организации с DevOps стремятся автоматизировать непрерывное развертывание, чтобы все происходило гладко и надежно. Компании, использующие методы DevOps, получают выгоду за счет значительного сокращения времени выхода на рынок и повышения удовлетворенности клиентов, качества продукции, а также производительности и эффективности сотрудников. Минусы: некоторым клиентам не нужны постоянные обновления своих систем; в некоторых отраслях есть правила, которые требуют всестороннего тестирования, прежде чем проект может перейти в операционную фазу; если разные отделы используют разные среды, необнаруженные проблемы могут перейти в производственную среду; некоторые атрибуты качества требуют вмешательства человека, что замедляет конвейер доставки.

Многие считают метод водопада самым традиционным методом разработки программного обеспечения. Каскадный метод — это жесткая линейная модель, состоящая из последовательных этапов (требования, проектирование, реализация, проверка, обслуживание), ориентированных на определенные цели. Каждый этап должен быть завершен на 100%, прежде чем можно будет начать следующий этап. Обычно нет процесса возврата, чтобы изменить проект или направление. Плюсы: линейный характер метода разработки водопада упрощает понимание и управление; в проектах с четкими целями и стабильными требованиями лучше всего использовать метод водопада; менее опытные руководители проектов и проектные группы, а также команды, состав которых часто меняется, могут получить наибольшую выгоду от использования методологии разработки водопада. Минусы: метод создания водопада часто медленный и дорогостоящий из-за его жесткой структуры и жесткого контроля; эти недостатки могут побудить пользователей метода водопада к изучению других методологий разработки программного обеспечения.

Быстрая разработка приложений (RAD) — это сжатый процесс разработки, в результате которого создается высококачественная система с низкими инвестиционными затратами. Скотт Стинер, генеральный директор и президент UM Technologies, сказал в Forbes: «Процесс RAD позволяет нашим разработчикам быстро приспосабливаться к меняющимся требованиям на быстро меняющемся и постоянно меняющемся рынке». Возможность быстрой настройки — это то, что обеспечивает такие низкие инвестиционные затраты. Метод быстрой разработки приложений состоит из четырех этапов: планирование требований, пользовательское проектирование, создание и переключение. Этапы пользовательского проектирования и строительства повторяются до тех пор, пока пользователь не подтвердит, что продукт соответствует всем требованиям. Плюсы: быстрая разработка приложений наиболее эффективна для проектов с четко определенной бизнес-целью и четко определенной группой пользователей, которые не требуют вычислительной сложности. RAD особенно полезен для малых и средних проектов, которые зависят от времени. Минусы: для быстрой разработки приложений требуется стабильный состав команды с высококвалифицированными разработчиками и пользователями, хорошо разбирающимися в области приложения. Глубокие знания необходимы для сжатого графика разработки, который требует утверждения после каждого этапа разработки. Организации, не отвечающие этим требованиям, вряд ли получат выгоду от RAD.[5][11]

## 1.3. Инструменты и фреймворки гибкой методологии разработки

Начнем со Scrum, так как это самый распространенный и самый плохо реализованный из фреймворков Agile. Официально из руководства по Scrum: Scrum состоит из 3 ролей (Скрам-мастер, владелец продукта и команда разработчиков), 5 событий (Sprint, Sprint Planning, Daily Scrum, Sprint Review и Sprint Retrospective) и 3 артефактов (бэклог проекта, журнал пожеланий спринта и инкремент продукта). Scrum идеально подходит для проектов по разработке программного обеспечения, требования к которым меняются. А также для: сложной работы, требующей дисциплины и навыков; отраслей, сталкивающихся с меняющимися рынками и сложной конкуренцией при разработке продуктов или программного обеспечения; организаций, которым для достижения успеха необходимо произвести более интенсивные или серьёзные изменения в своей культуре или структуре. Если к проекту относится один или несколько элементов, то Scrum будет хорошим кандидатом в качестве Agile-фреймворка для внедрения. В обратном случае, стоит рассмотреть что-то другое.[13]

Теперь рассмотрим Канбан. Это самый легкий из фреймворков Agile, он довольно популярен, это самый быстрорастущий метод, тесно связанный с Lean. От создателя Дэвида Дж. Андерсона: Канбан состоит из трех основополагающих принципов: (1) начните с того, что вы делаете сейчас, (2) согласитесь проводить постепенные, эволюционные изменения и (3) уважайте текущий процесс, роли, обязанности и звания. Затем опирайтесь на это понимание с помощью пяти основных свойств: визуализируйте рабочий процесс; ограничить незавершенное производство; управление потоком; сделайте политики процессов явными; совершенствуйте вместе (используя модели и научный метод). Канбан идеально подходит для сложных систем или процессов, требующие длительные сроки выполнения заказа (недели и месяцы); для организаций, которые хотят небольших, регулярных и постепенных изменений в своей структуре или процессах; для аппаратного обеспечения, сложного продукта или программного обеспечения, которое необходимо улучшать в течение более длительных периодов времени.[7]

Обратимся к экстремальному программированию, так как это один из самых популярных фреймворков Agile, который более интенсивен, чем Scrum и Kanban. Экстремальное программирование (также известное как XP) имеет пять ценностей: (1) простота, (2) общение, (3) обратная связь, (4) уважение и (5) смелость.

XP также имеет множество правил в пяти областях, в том числе: планирование (пользовательские истории, планирование выпуска и т. д.), управление (открытое рабочее пространство, устойчивый темп и т. д.), дизайн (простота, карточки CRC и т. д.), кодирование (стандарты, парное программирование и др.), тестирование (модульное тестирование, приемочные тесты и т. д.). Экстремальное программирование идеально подходит для любого проекта разработки программного обеспечения. XP требует другого уровня поддержки как со стороны членов команды, так и со стороны среднего и высшего руководства. Для получения желаемых качественных и количественных преимуществ требуется гораздо больше дисциплины и инвестиций.[3]

# ВЫВОД К ГЛАВЕ I

Есть множество разных моделей и фреймворков для управления разработкой программного обеспечения. У каждого из них есть как плюсы, так и минусы. Для разработки программного проекта сервиса для управления задачами, который изначально будет разработан как минимально жизнеспособный продукт (MVP) и будет часто обновляться, подойдёт методология гибкой разработки и фреймворк Канбан.

# ГЛАВА II. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МЕТОДОЛОГИИ

## 2.1. Техническое задание

Техническое задание (ТЗ, техзадание) — документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями.[14]

При создании технического задания стоит опираться на два стандарта — ГОСТ 19.201-78 и ГОСТ 34.602-89, но так как в первом ГОСТе указано очень мало требований, в своей работе я опирался на ГОСТ 34.602-89.

Рассмотрим некоторые разделы данного ГОСТа. В разделе 2 «Состав и содержание», пункта 6, подпункта 1 указываются требования к системе в целом. Рассмотрим некоторые его части.[9]

Требования к структуре и функционированию системы: страница входа/регистрации аккаунта с вводом электронной почты и пароля (и опциональной кода 2ФА), основной сайт для создания и редактирование задач, настройки, сайт поддержки с FAQ о приложении и контактами для связи с разработчиками; есть поиск среди задач; есть возможность размещать задачи в проектах, которые можно создавать/изменять/удалять; проект написан на HTML, CSS и JavaScript; проект должен быть совместим со всеми популярными браузерами; проект модульный; есть открытый API для работы с другими приложениями/системами.

Показатели назначения: проект легко расширяем и изменяем.

Требования к надежности: аптайм 99,9 %, исправление критических ошибок за 2 дня.

Требования к эргономике и технической эстетике: сайт оптимизирован как для десктопа, так и для мобильных устройств, шрифт должен быть легко читаемым и контрастным, поддержка тёмной и светлой темы.

Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы: сервера арендуются у крупного провайдера с автоматическим обслуживанием.

Требования к защите информации от несанкционированного доступа: использование RSA-2048, AES-256, HTTPS-соединение, хэширование паролей пользователей, 2ФА для пользователей, отсутствие доступа к личной информации у разработчика.

## 2.2. Диаграмма Ганта

После составления технического задания стоит сделать диаграмму Ганта для понимая, сколько времени уйдёт на разные этапы создания приложения. Важно рационально распределить время.

Диаграмма Ганта — это горизонтальная столбчатая диаграмма с временной шкалой, которая используется для иллюстрации плана работ по проекту с привязкой ко времени.[15]

Чтобы создать диаграмму Ганта, мной было использовано онлайн-приложение GanttPRO, так как его интерфейс интуитивно понятен, оно популярное, есть обширная справка и есть бесплатная версия для студентов.

Для начала нужно ввести задачи: составление ТЗ, проектирование страниц, разработка дизайна страниц, вёрстка страниц, программирование бэкенда и фронтенда сайта, тестирование сайта, развёртывание сайта.

Затем нужно выбрать промежутки их выполнения, изменить цвета для наглядности. Масштаб был выбран недельный для упрощения представления. Полученная диаграмма представлена на рисунке 1.

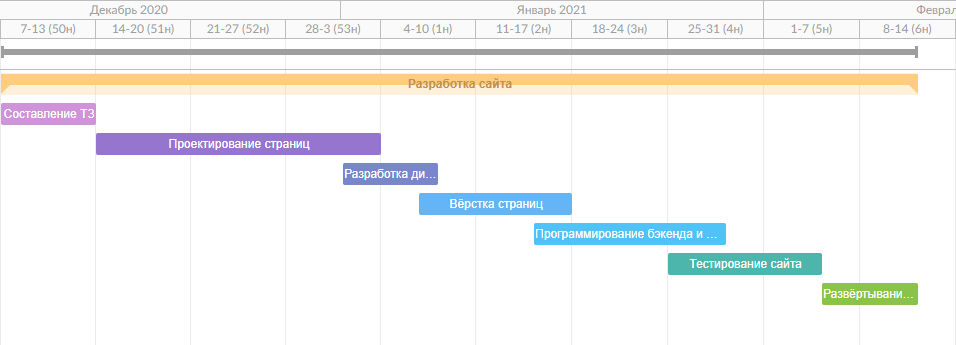


Рисунок 1

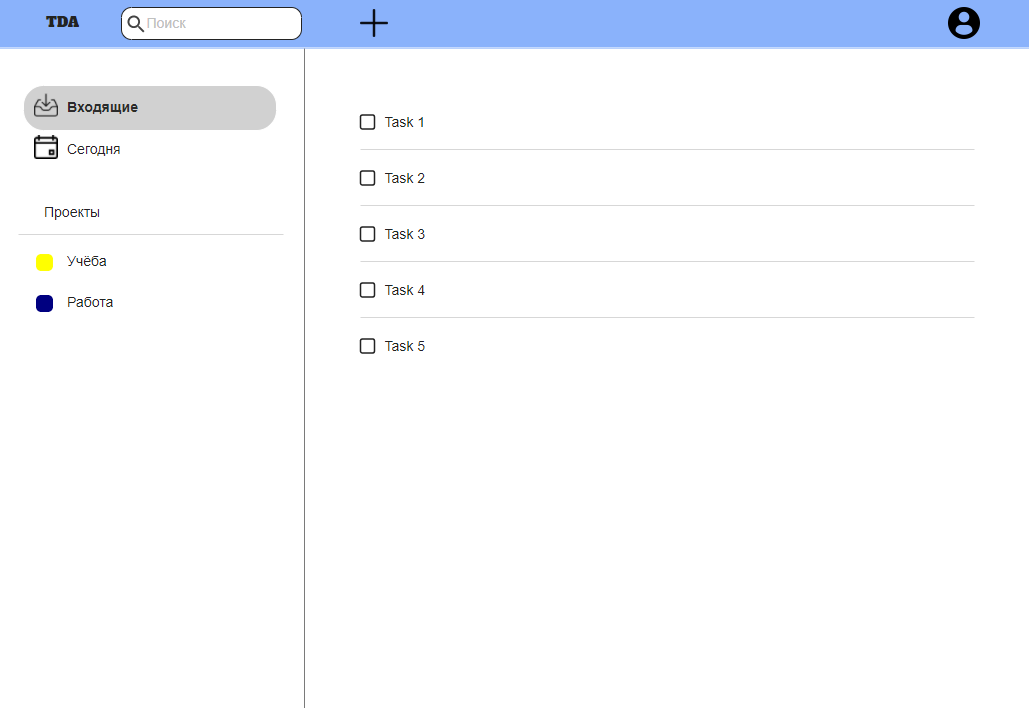
## 2.3. Макет

После того, как было распределено время, можно перейти к созданию макета.

Дизайн-макет сайта — это визуальный образ будущего сайта, разработанный с учетом технических возможностей HTML верстки. Такой макет является демонстрацией того, как визуально будет выглядеть сайт после верстки и наполнения.[12]

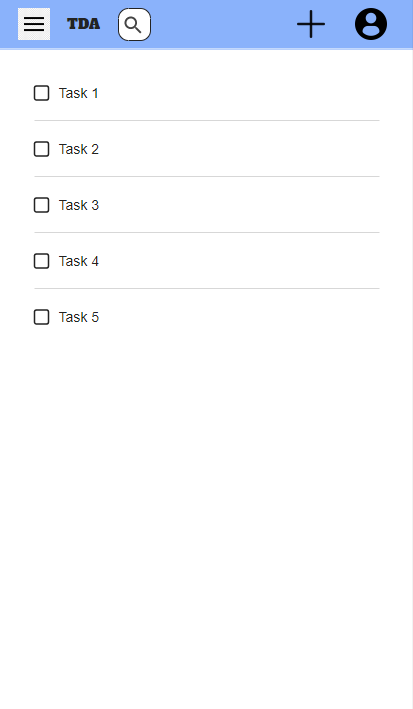
Макет был создан в сервисе Moqups, так как он в бесплатной версии имеет достаточной функционал для создания необходимого в моём проекте макета — множество инструментов и форм, создание нескольких страниц для разных версий сайта и готовые шаблоны.

В хедере (шапке) была размещена иконка сайта, ведущая на главную страницу сайта, далее поиск среди задач, затем кнопка добавления задач и иконка профиля, содержащая кнопку перехода в настройки и другую информацию о приложении. В основной части сайта находится список задач, а слева находится список проектов — предустановленные «Входящие» и «Сегодня», а также проекты, которые пользователь сам может создать. В качестве примера введены «Учёба» и «Работа». Полученный макет представлен на рисунке 2.



Рисунок

Затем был разработан макет мобильной версии, в котором список проектов был скрыт с основного экрана. Так как иначе список задач пришлось бы делать очень узким, и из-за этого с ними было сложно работать. Чтобы открыть список проектов в мобильной версии, нужно нажать на иконку «гамбургерного» меню. Полученный макет представлен на рисунке 3.



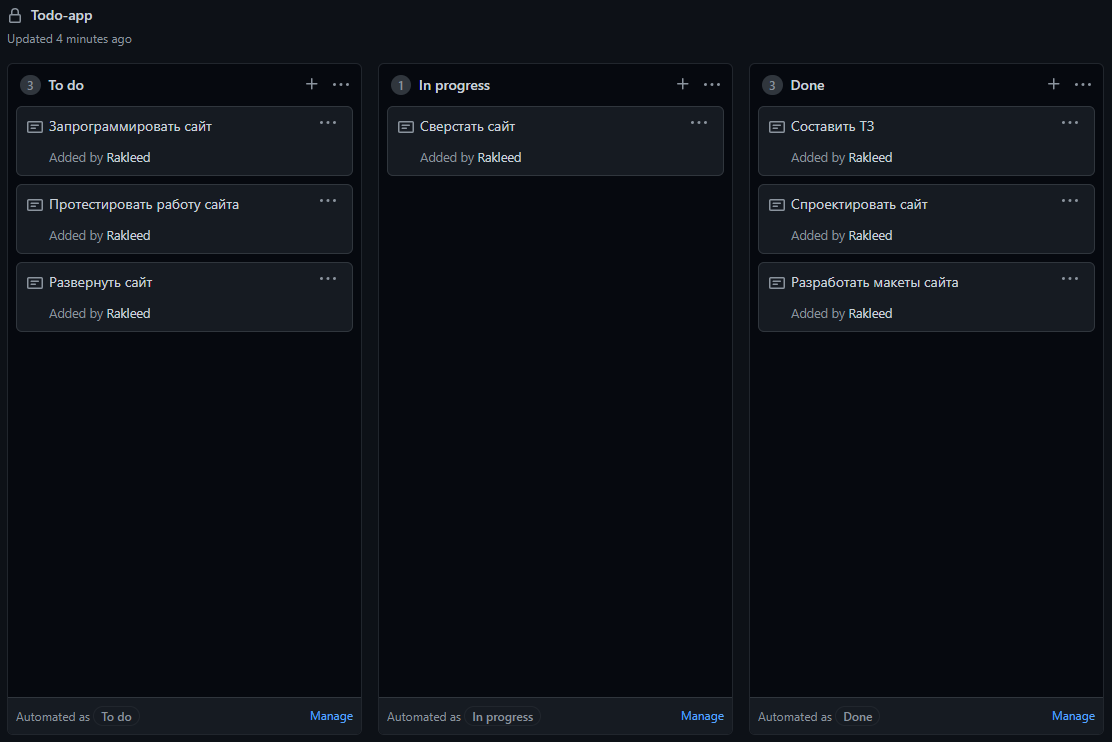
Рисунок

## 2.4. Инструмент управления проектом

Чтобы эффективно разрабатывать проект, нужно было выбрать систему управления проектом.

Git был выбран в качестве системы управления версиями, так как это самая популярная и удобная система для управления кодом. Код приложения было решено хранить в сервисе GitHub, так как это самое популярное, удобное хранилище кода, по которому есть много справочной информации.

Соответственно для управления проектом был выбран GitHub Projects, так как он бесплатный и удобно, что код и система управления проектом находятся в одном месте. Например, из-за этого можно легко добавлять issue и pull request в виде карточек на доску. На рисунке 4 представлен пример частично заполненной доски по моему проекту.[4]



Рисунок

## 2.5. UML-диаграмма

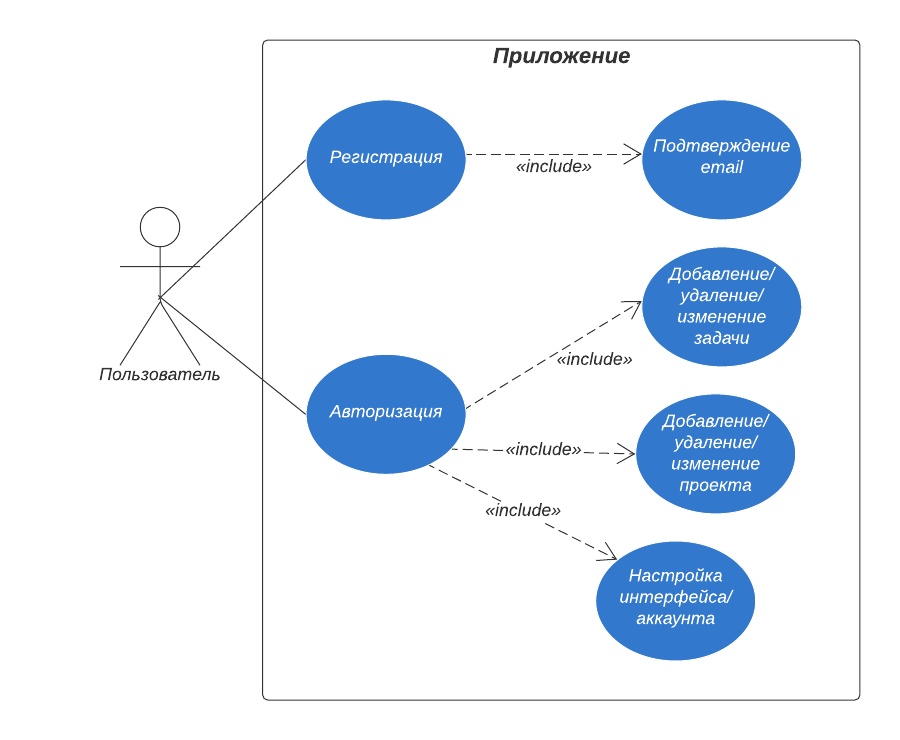
Теперь чтобы визуализировать часть архитектуры приложения, создадим диаграмму прецендентов.

Диаграмма вариантов использования в UML (или диаграмма прецендентов) — это основная форма требований к системе/программному обеспечению для новой, недостаточно развитой программы. Сценарии использования определяют ожидаемое поведение (что), а не точный способ его реализации (как). Ключевой концепцией моделирования вариантов использования является то, что она помогает нам проектировать систему с точки зрения конечного пользователя. Это эффективный метод передачи поведения системы в терминах пользователя путем определения всего внешнего видимого поведения системы.[8]

Актор — стилизованный человечек, обозначающий набор ролей пользователя (понимается в широком смысле: человек, внешняя сущность, класс, другая система), взаимодействующего с некоторой сущностью (системой, подсистемой, классом). Прецедент — эллипс с надписью, обозначающий выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым акторами результатам.[10]

Для составления UML-диаграммы был использован сервис Lucidchart, так как у него есть бесплатный тариф, дружелюбный интерфейс, множество инструментов и шаблонов.

На диаграмме в качестве актора используется пользователь приложения, который может зарегистрироваться или авторизоваться, и после этого уже использовать полноценно систему. На рисунке 5 представлена такая диаграмма прецендентов.



Рисунок

## 2.6. Тестирование

Так как уже многое прояснилось и имеется достаточно данных о проекте, можно спланировать юзабилити-тестирование.

Юзабилити-тестирование — это метод, используемый в ориентированном на пользователя дизайне взаимодействия для оценки продукта путем тестирования его на пользователях. Его можно рассматривать как незаменимую практику, поскольку она дает информацию о том, как реальные пользователи используют систему.[6]

Метод карточной сортировки является быстрым, недорогим и надежным методом, который помогает продумать общую структуру и принципы размещения информации на сайте, а также некоторые вопросы, касающиеся навигации, организации и т. п.

Будут собраны потенциальные пользователи на форумах, они получат особый набор карточек, на каждой из которых написан определенный элемент содержимого сайта. Затем участники должны сгруппировать эти карточки таким образом, который покажется им наиболее очевидным и логичным. Результат этого процесса поможет понять, как потенциальные пользователи структурируют информацию у себя в голове, и в какой форме они ожидают увидеть её на сайте.

Смешанный метод карточной сортировки не предназначен для оценки существующих интерфейсов, поэтому он подходит в моём случае.

# ВЫВОД К ГЛАВЕ II

Была представлена часть технического задания с описанием некоторых характеристик приложения. Затем была продемонстрирована диаграмма Ганта для понимая того, сколько времени может понадобится на различны этапах создания приложения. Далее был показан макет приложения для представления примерного видения интерфейса приложения. Также в качестве инструмента управления проектом был выбран GitHub Projects из-за удобства и полноценной интеграции со средством хранения и управления кодом GitHub. Чтобы визуализировать часть архитектуры приложения и понять пути взаимодействия пользователя с приложением была создана UML-диаграмма в сервисе Lucidchart. В конце было описано юзабилити-тестирование, которое пока не было произведено.

# ЗАКЛЮЧЕНИЕ

В ходе работы были описаны различные парадигмы и модели разработки программного обеспечения, также были описаны инструменты и фреймворки гибкой методологии разработки. Были указаны их достоинства и недостатки, в каком случае что лучше использовать. В результате анализа была выбрана гибкая модель разработки и фреймворк Канбан как наиболее подходящие для разработки программного проекта сервиса для управления задачами, который изначально будет разработан как минимально жизнеспособный продукт (MVP) и будет часто обновляться.

В практической части была написана часть технического задания, составлена диаграмма Ганта в сервисе GanttPRO, представлены макеты десктопной и мобильной версии проекта с помощью сервиса Moqups, выбран инструмент управления проектом GitHub Projects, составлена UML-диаграмма прецендентов в сервисе Lucidchart и описано то, как можно провести юзабилити-тестирование.

Проект готов к дальнейшей разработке приложения для управления задачами.

# ЛИТЕРАТУРА

1. Agile. Оценка и планирование проектов / Кон М.; Пер. с англ. — М.: Альпина Паблишер, 2018.
2. Agile-менеджмент. Лидерство и управление командами / Аппело Ю., Пер. с англ. — М.: Альпина Паблишер, 2019.
3. Choosing the Right Agile Framework for You // LinkedIn URL: https://www.linkedin.com/pulse/choosing-right-agile-framework-you-paul-j-heidema (дата обращения: 01.12.2020).
4. Features • Project Management // GitHub URL: https://github.com/features/project-management/ (дата обращения: 11.12.2020).
5. Top 4 software development methodologies // Synopsys URL: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/ (дата обращения: 02.12.2020).
6. Usability testing // Wikipedia URL: https://en.wikipedia.org/wiki/Usability\_testing (дата обращения: 14.12.2020).
7. What is kanban? // Atlassian Agile Coach URL: https://www.atlassian.com/agile/kanban (дата обращения: 04.12.2020).
8. What is Use Case Diagram? // Visual Paradigm URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата обращения: 10.12.2020).
9. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы // Электронный фонд правовой и нормативно-технической документации URL: http://docs.cntd.ru/document/gost-34-602-89 (дата обращения: 10.12.2020).
10. Диаграмма прецедентов // Википедия URL: https://ru.wikipedia.org/wiki/Диаграмма\_прецедентов (дата обращения: 13.12.2020).
11. Проектирование информационных систем / Чистов Д. В., Мельников П. П., Золотарюк А. В., Ничепорук Н. Б.; под ред. Чистов Д. В. — М.: Юрайт, 2020.
12. Разработка дизайн-макета и создание структуры сайта // OKsofr URL: https://oksoft.ru/create\_site\_design (дата обращения: 09.12.2020).
13. Сравнение Kanban и Scrum // Atlassian URL: https://www.atlassian.com/ru/agile/kanban/kanban-vs-scrum (дата обращения: 17.12.2020).
14. Техническое задание // Википедия URL: https://ru.wikipedia.org/wiki/Техническое\_задание (дата обращения: 05.12.2020).
15. Что такое диаграмма Ганта? // Atlassian Agile Coach URL: https://www.atlassian.com/ru/agile/project-management/gantt-chart (дата обращения: 07.12.2020).