

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Заполярный Государственный Университет им. Н.М. Федоровского»**

**Информационные технологии**

**Лабораторная работа №1**

**Выполнил студент группы ИС-23**

**Киселёв Иван Михайлович**

Оглавление

[**Введение** 3](#_Toc212816663)

[**Основная часть** 4](#_Toc212816664)

[**1. Регистрация и настройка инструментов** 4](#_Toc212816665)

[**2. Управление задачами** 7](#_Toc212816666)

[**3. Методологии и их анализ** 9](#_Toc212816667)

[**4. Управление рисками в IT-проектах** 12](#_Toc212816668)

[**Заключение** 16](#_Toc212816669)

# **Введение**

**Целью** данной лабораторной работы является освоение методов управления проектом с использованием инструментов визуального планирования и гибких методологий. В рамках работы реализуется проект «Компьютерный клуб», включающий постановку задач, их структурирование, распределение по этапам и симуляцию выполнения.

**Задачи работы:**

* зарегистрировать проект в системе управления задачами;
* выбрать, описать и обосновать методологию управления;
* сформировать набор задач, определить приоритеты и сроки;
* настроить рабочее пространство и провести симуляцию выполнения деятельности в рамках условного проекта;
* проанализировать выбранную методологию и сравнить её с другими.

# **Основная часть**

## **1. Регистрация и настройка инструментов**

Для управления проектом «Компьютерный клуб» была выбрана платформа **Trello** – облачный инструмент визуального управления задачами, поддерживающий гибкую настройку рабочих процессов и интеграцию с другими сервисами разработки.

В качестве методологии управления проектом принята **Kanban**. Выбор обусловлен её ориентацией на непрерывный поток задач, визуализацию рабочего процесса и возможность оперативной адаптации к изменениям – что соответствует характеру учебного проекта с динамически уточняемыми требованиями.

В Trello создан проект под названием «Computer Club», структурированный в соответствии с принципами Kanban. Доска включает следующие колонки (статусы задач):

* Backlog
* To Do
* In Progress
* Review
* Done

План проекта охватывает пять основных этапов жизненного цикла:

1. Инициация – анализ предметной области, формирование требований, подготовка технического задания.
2. Планирование – выбор стека технологий, проектирование архитектуры, моделирование бизнес-процессов (IDEF0, DFD).
3. Разработка – реализация клиентской и серверной частей, базы данных, API, пользовательского интерфейса.
4. Тестирование – проверка работоспособности компонентов, интеграционное и пользовательское тестирование.
5. Развертывание – настройка CI/CD, подготовка к эксплуатации, финальная верификация.

Сформирован набор из **19** задач:

Таблица 1. Задачи, необходимые для реализации проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Задача | Список | Приоритет | Срок выполнения |
| 1 | Разработка бэк-части проекта WEB-сайта | BACKLOG | - | - |
| 2 | Разработка фронт-части проекта WEB-сайта | BACKLOG | - | - |
| 3 | Создание связи между программой и сайтом (связь back-ов между собой) | BACKLOG | - | - |
| 4 | Проведение тестирования работоспособности MVP-версии продукта | TO DO | Средний | - |
| 5 | Проведение финального тестирование готового продукта | TO DO | Средний | - |
| 6 | Реализация модуля авторизации | TO DO | Низкий | - |
| 7 | Реализация рабочей связи "Клиент-Сервер" | TO DO | Высокий | - |
| 8 | Настройка CI/CD пайплайна для автоматической сборки и деплоя | TO DO | Низкий | - |
| 9 | Тестирование работоспособности интерфейса UI/UX | TO DO | Низкий | - |
| 10 | Написание серверной части программы "Computer Club" | IN PROGRESS | Высокий | 13.10.25 – 03.11.25 |
| 11 | Написание клиентской части программы "Computer Club" | IN PROGRESS | Высокий | 13.10.25 – 26.10.25 |
| 12 | Реализация API для проверки статуса занятости компьютера на текущий момент времени | IN PROGRESS | Средний | 16.10.25 – 30.10.25 |
| 13 | Реализация базы данных для проекта (PostgreSQL) | REVIEW | Высокий | 05.10.25 – 19.10.25 |
| 14 | Написание технической документации API | REVIEW | Средний | 11.10.25 – 18.10.25 |
| 15 | Работа над дизайном пользовательского интерфейса клиентской части ПО "Computer Club" | REVIEW | Низкий | 12.10.25 – 19.10.25 |
| 16 | Подготовить ТЗ программного продукта | DONE | Высокий | 06.10.25 – 13.10.25 |
| 17 | Настройка и подготовка системы контроля версий (Git + GitHub) | DONE | Средний | 07.10.25 – 10.10.25 |
| 18 | Провести системный анализ методологией IDEF0 | DONE | Средний | 01.10.25 – 08.10.25 |
| 19 | Провести системный анализ методологией DFD | DONE | Средний | 24.09.25 – 01.10.25 |

**Приоритеты** установлены по шкале:

* Высокий – критически важные для MVP задачи;
* Средний – вспомогательные и малозатратные задачи;
* Низкий – задачи дополнительного характера, необязательные для MVP-продукта.

**Сроки** назначены с учётом логической последовательности выполнения и условно-установленной «текущей» датой – 16.10.2025.

## **2. Управление задачами**

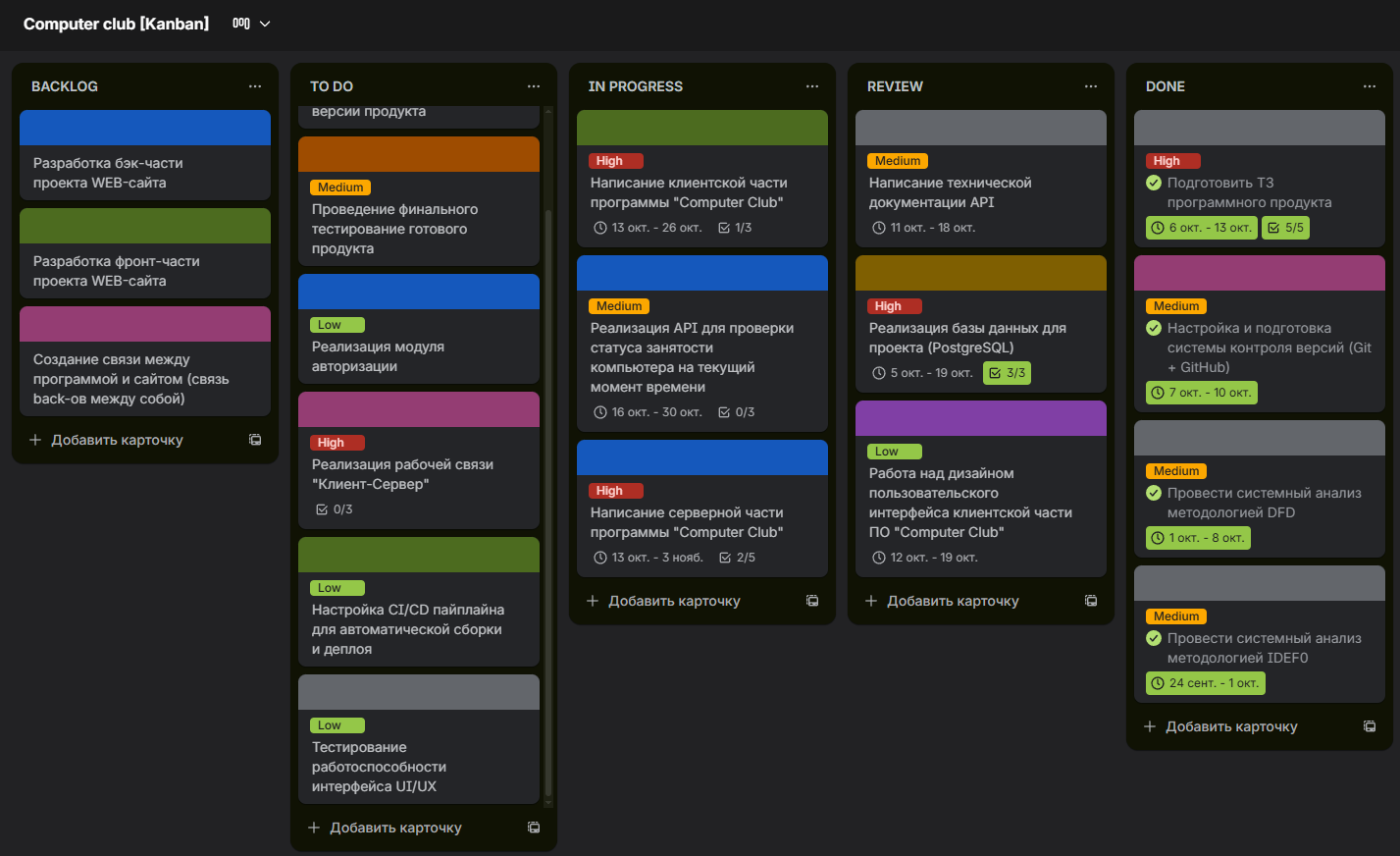


Рисунок 1. Доска Kanban в рабочем пространстве «Computer club»

Проведена имитация активной разработки: задачи последовательно перемещались по колонкам по мере выполнения.

Для ограничения перегрузки и повышения фокусировки на завершении текущих работ был введён лимит WIP (Work in Progress):

* максимум 3 задачи одновременно в колонке In Progress;
* максимум 3 задачи в колонке Review.

Это позволило избежать ситуации, когда разработчик «берёт много задач, но ничего не завершает» – ключевой принцип эффективного Kanban.

На доске активированы две автоматизации для поддержания баланса рабочего потока в соответствии с принципами Kanban:

1. Правило №1:
   * Условие: когда в колонке «IN PROGRESS» находится более 3 карточек;
   * Действие: переместить новую карточку в начало колонки “TO DO”;
   * Цель: предотвратить перегрузку текущих задач и обеспечить соблюдение лимита WIP (Work in Progress).
2. Правило №2:
   * Условие: когда в колонке “REVIEW” находится более 3 карточек;
   * Действие: переместить новую карточку в начало колонки «IN PROGRESS»;
   * Цель: предотвратить перегрузку текущих задач и обеспечить соблюдение лимита WIP (Work in Progress).

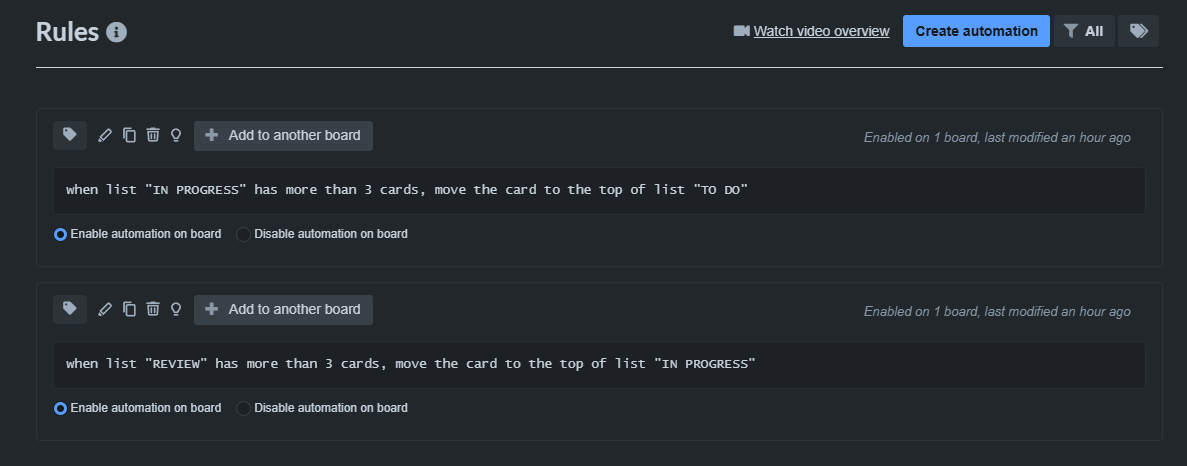


Рисунок 2. Установленные правила в рабочем пространстве «Computer club»

## **3. Методологии и их анализ**

Для управления проектом «Компьютерный клуб» была выбрана методология **Kanban** – гибкий подход к управлению работой, основанный на принципах визуализации, ограничения незавершённой работы (WIP) и непрерывного улучшения процесса.

Kanban не предполагает фиксированных временных итераций, ролей или церемоний. Вместо этого он фокусируется на реальном потоке задач: каждая задача проходит через последовательность статусов («To Do» 🡪 «In Progress» 🡪 «Done»), отображаемых на доске. Основные элементы методологии:

* Визуальная доска – отражает текущее состояние всех задач.
* Лимиты WIP – ограничивают количество задач на каждом этапе, чтобы избежать перегрузки и ускорить завершение.
* Непрерывный поток – новые задачи могут добавляться в любой момент, приоритеты могут меняться динамически.
* Метрики – такие как цикл времени и время ожидания, помогают оценивать эффективность.

Kanban особенно эффективен в условиях нестабильных требований, малых команд или индивидуальной разработки, где важна гибкость, а не жёсткая дисциплина итераций.

**Scrum и Waterfall:**

1. Scrum основан на итеративной разработке с короткими циклами –спринтами, в то время как Waterfall предполагает строго последовательное выполнение фаз без возвратов.
2. В Scrum требования могут уточняться на протяжении всего проекта, а в Waterfall они фиксируются на старте и практически не подлежат изменению.
3. Тестирование в Scrum интегрировано в каждый спринт, тогда как в Waterfall оно проводится один раз – в конце. Это делает Scrum значительно более адаптивным, а Waterfall – предсказуемым, но рискованным при нестабильных требованиях.

**Scrum и Kanban:**

1. Обе методологии гибкие, но Scrum структурирует работу во времени (фиксированные спринты, роли, церемонии), а Kanban – в потоке (непрерывное выполнение задач без итераций).
2. В Scrum состав задач на спринт не меняется, в Kanban – приоритеты можно корректировать в любой момент.
3. Scrum лучше подходит для команд с регулярной синхронизацией, Kanban – для малых групп или индивидуальной работы, где важна оперативность и минимум формальностей.

**Плюсы и минусы методологий и подходов:**

**Waterfall**

* Плюсы:
  + простота планирования и отчётности;
  + чёткая документация на каждом этапе;
  + подходит для проектов с жёсткими регуляторными требованиями.
* Минусы:
  + невозможность быстро реагировать на изменения;
  + высокий риск провала из-за позднего тестирования;
  + низкая вовлечённость заказчика до финальной стадии.

**Scrum**

* Плюсы:
  + регулярная поставка рабочего продукта;
  + прозрачность и вовлечённость за счёт демо и ретроспектив;
  + хорошо масштабируется в командах.
* Минусы:
  + требует дисциплины и понимания ролей;
  + неэффективен при частых внешних прерываниях;
  + избыточен для малых или индивидуальных проектов.

**Kanban**

* Плюсы:
  + минимальный порог входа;
  + гибкость в управлении приоритетами;
  + лёгкость визуализации узких мест через WIP-лимиты;
  + подходит для смешанной нагрузки.
* Минусы:
  + отсутствие временных рамок может снижать дисциплину;
  + менее структурирован для новичков без опыта;
  + требует самоконтроля при отсутствии команды.

## **4. Управление рисками в IT-проектах**

Управление рисками является неотъемлемой частью успешной реализации ИТ-проектов, особенно в сфере разработки программного обеспечения, где высокая степень неопределённости, быстро меняющиеся требования и технологические вызовы создают значительные угрозы для сроков, бюджета и качества продукта. Эффективное управление рисками позволяет своевременно выявлять потенциальные угрозы, оценивать их влияние и разрабатывать стратегии по их минимизации или устранению.

**Основные подходы к управлению рисками в проектах по разработке ПО:**

1. Идентификация рисков
   * Проводится на ранних этапах проекта с использованием методов: мозгового штурма, анализа чек-листов, интервью с экспертами, анализа аналогичных проектов.
   * Типичные категории рисков: технические, организационные, внешние, связанные с требованиями, персоналом и ресурсами.
2. Анализ и оценка рисков
   * Качественная оценка: определение вероятности наступления и потенциального воздействия.
   * Количественная оценка: моделирование последствий, расчёт ожидаемой денежной стоимости.
3. Планирование реагирования на риски
   * Стратегии: избежание, передача (аутсорсинг), снижение, принятие.
   * Разработка резервных планов и «триггеров» активации мер реагирования.
4. Мониторинг и контроль рисков
   * Регулярный пересмотр реестра рисков.
   * Использование KPI и метрик для отслеживания изменений в уровне рисков.
   * Интеграция управления рисками в повседневную деятельность команды.

**Влияние методологий разработки на управление рисками**

**Waterfall (Каскадная модель):**

Помогает:

1. Раннее планирование рисков: на этапе анализа требований и проектирования можно заранее составить полный реестр рисков, особенно технических и организационных.
2. Чёткая документация: позволяет фиксировать предположения, зависимости и ограничения, что снижает риск недопонимания.
3. Предсказуемость: подходит для проектов с регуляторными требованиями, где необходимо строгое соблюдение процессов.

Мешает:

1. Жёсткая последовательность этапов: если риск выявлен на этапе тестирования (например, несоответствие требований), возврат к проектированию или анализу требований крайне затратен.
2. Отсутствие обратной связи до конца проекта: риск «неправильного продукта» может быть обнаружен слишком поздно.
3. Низкая адаптивность: невозможно быстро реагировать на внешние изменения (смена технологий, уход ключевого специалиста).

**Agile (Scrum, Kanban)**

Помогает:

1. Итеративная доставка ценности: риски выявляются уже в первые спринты (сложность интеграции API, нестабильность требований).
2. Регулярная обратная связь: разработчик и заказчик участвуют постоянно, что снижает риск разработки ненужного функционала.
3. Гибкость приоритизации: команда может перенаправить усилия на устранение критических рисков (технический долг) в следующем спринте.

Мешает:

1. Недостаточное долгосрочное планирование: стратегические риски (масштабируемость, соответствие архитектуры будущим потребностям) могут быть проигнорированы из-за фокуса на коротких циклах.
2. Зависимость от команды: если в команде нет эксперта по безопасности или архитектуре, соответствующие риски могут не быть выявлены.
3. Слабая документация: при смене состава команды возникает риск потери знаний и повторения ошибок.

**Примеры реальных рисков и методов их минимизации по ключевым этапам ИТ-проекта**

**1. Непонимание или изменение требований в ходе проекта**

Часто заказчик не может точно сформулировать свои ожидания на старте, либо меняет их по мере развития проекта. Это ведёт к разработке ненужного функционала, переработкам и срыву сроков.

Минимизация: активное вовлечение заказчика, использование прототипов и демоверсий на ранних этапах, работа с гибкими требованиями в виде реализаций методологий с чёткими критериями приёмки. В Waterfall – строгая фиксация требований и управление изменениями через формальный процесс.

**2. Технические риски: неудачный выбор архитектуры или технологий**

Неправильно выбранная архитектура или устаревший стек технологий могут привести к невозможности масштабировать систему, высокой стоимости поддержки или даже провалу проекта.

Минимизация: проведение мероприятий-исследований на этапе проектирования, привлечение архитекторов и технических экспертов, поэтапная эволюция архитектуры с регулярным рефакторингом (Agile).

**3. Низкое качество кода и накопление технического долга**

Под давлением сроков команда может жертвовать качеством кода, что со временем замедляет разработку, увеличивает число ошибок и усложняет поддержку.

Минимизация: внедрение практик инженерного качества – «code review», автоматическое тестирование, статический анализ кода. В Agile технический долг явно учитывается в бэклоге и регулярно «погашается» через выделение спринтовых задач на рефакторинг.

**4. Проблемы при развёртывании и эксплуатации**

Даже полностью протестированное решение может дать сбой в конечных этапах разработки из-за различий окружений, недостаточной подготовки инфраструктуры или отсутствия мониторинга.

Минимизация: использование заготовленной среды, идентичной конечному продукту, автоматизация развёртывания (CI/CD), постепенные релизы, наличие чёткого плана отката и системы логирования/мониторинга.

# **Заключение**

В ходе лабораторной работы был успешно реализован проект «Computer club» с применением методологии Kanban и инструмента Trello. Выполнена регистрация и настройка доски, сформирован план проекта, распределены задачи с учётом приоритетов и сроков, а также проведена симуляция рабочего процесса и настроены правила рабочей области. Анализ показал, что Kanban обеспечивает достаточную гибкость и прозрачность для управления учебным проектом, особенно в условиях индивидуальной разработки. Все поставленные цели и задачи работы успешно выполнены.