# Лабораторная работа № 1. Планирование работ по выполнению проекта «Разработка и внедрение программного обеспечения»

**Цели занятия:** Получение первичных навыков планирования работ по разработке и внедрению автоматизированных информационных систем на основе распространенных моделей жизненных циклов программных продуктов.

**Цель работы:** на основе технического задания на разработку и внедрение автоматизированной информационной системы сформировать календарный план выполнения работ по проекту.

**Средства реализации:** Ограничений не накладывается.

**Исходные данные:** в качестве исходных данных выступает техническое задание на разработку и внедрение автоматизированной информационной системы, перечень доступных для использования трудовых ресурсов.

## Выполнение работы

Для выполнения данной лабораторной работы используется разработанный материал курсовой работы по дисциплине проектирование информационных систем на тему «Проектирование информационной системы для учёта спортивных мероприятий ВУЗа».

Цель исследования: разработка проекта информационной системы для учета спортивных мероприятий вуза.

Задачи исследования представлены ниже:

* изучить текущие процессы организации и проведения спортивных мероприятий в вузе;
* выявление проблем и недостатков в существующей системе учета спортивных мероприятий;
* изучить методы и средства проектирования и средства проектирования информационных систем;
* подготовить рекомендации по улучшению системы учета спортивных мероприятий на основе проведенного анализа;
* спроектировать диаграммы IDEF0, DFD, IDEF3, IDEF1X;

Существует три основные модели жизненного цикла разработки программного обеспечения:

* Каскадная модель: В этой модели процесс разработки разбивается на последовательные этапы (анализ требований, проектирование, реализация, тестирование, внедрение). Переход к следующему этапу происходит после полного завершения предыдущего этапа.
  + Плюсы: простота и предсказуемость.
  + Минусы: неспособность адаптироваться к изменениям и отсутствие обратной связи между этапами.
* Итеративная модель: Это гибкая модель, предполагающая циклический процесс разработки с повторяющимися этапами. На каждом цикле производится анализ требований, проектирование и реализация. По мере выполнения каждого цикла вносятся изменения и улучшения в проект.
  + Плюсы: возможность быстро реагировать на изменения требований и наличие обратной связи между этапами.
  + Минусы: сложность управления проектом и необходимость высокой квалификации разработчиков.
* Спиральная модель: Эта модель объединяет преимущества каскадной и итеративной моделей. Она предполагает создание прототипов системы на разных этапах разработки. Это позволяет оценить риски и принять решения о продолжении или прекращении проекта.
  + Плюсы: раннее обнаружение рисков и возможность внесения изменений.
  + Минусы: увеличение времени на разработку проекта и потребность в опытных специалистах.

Для данной работы была выбрана именно каскадная модель. Так как она является одной из самых простых в понимании и использовании. Эта модель хорошо подходит для проектов с четкими требованиями и небольшим количеством изменений.

При использовании каскадной модели основными фазами будут являться анализ, проектирование, реализация, тестирование, внедрение (см. Рисунок 1)

Рисунок 1. Описание основных этапов каскадной модели

На каждом этапе можно выделяться основные задачи:

* Этап «Анализ»:
  + Сбор и документирование требований заказчика;
  + Анализ требований и их проверка на полноту, непротиворечивость и выполнимость;
  + Определение ограничений системы и интерфейсов с другими системами.
* Этап «Проектирование»:
  + Создание логической и физической моделей данных;
  + Разработка архитектуры и определение компонентов системы;
  + Проектирование и описание интерфейсов между компонентами системы;
  + Разработка процедур и алгоритмов обработки данных.
* Этап «Реализация»:
  + Написание и отладка кода программы;
  + Создание модульных и интеграционных тестов;
  + Внедрение необходимых библиотек и инструментов;
  + Разработка пользовательского интерфейса.
* Этап «Тестирование»:
  + Разработка и выполнение тестовых сценариев;
  + Выявление и устранение дефектов;
  + Оценка качества и полноты тестирования.
* Этап «Внедрение»
  + Установка и настройка системы на целевой платформе;
  + Обучение пользователей и администраторов;
  + Подготовка документации и инструкций для пользователей;
  + Проведение приемочных испытаний и демонстрация работы системы заказчику;
  + Поддержка пользователей и решение возникающих проблем.

В каскадной модели можно выделить что каждый следующий этап зависит от результата выполнения задач предыдущего.

В каскадной модели зависимости между задачами обычно определяются следующим образом:

* Сбор и документирование требований заказчика — это начальная задача, от которой зависят все остальные.
* Анализ требований и их проверка зависят от процесса сбора требований.
* Определение ограничений системы и интерфейсов зависит от процесса анализа требований.
* Создание логической и физической моделей данных, разработка архитектуры и определение компонентов системы, проектирование и описание интерфейсов, а также разработка процедур и алгоритмов обработки данных - все эти задачи зависят от определения ограничений системы и интерфейсов.
* Написание и отладка кода, создание модульных и интеграционных тестов, внедрение библиотек и инструментов, а также верстка пользовательского интерфейса зависят от процессов проектирования и разработки.
* Разработка тестовых сценариев, выявление и устранение дефектов, оценка качества и полноты тестирования зависят от процесса реализации.
* Установка и настройка системы, обучение пользователей и администраторов, подготовка документации и проведение приемочных испытаний зависят от процесса тестирования и результатов процесса реализации.
* Исправление ошибок, выпуск обновлений и патчей, поддержка пользователей и решение проблем зависят от этапа внедрения.

В каскадной модели задачи обязаны следовать в строгой последовательности:

1. Сбор и документирование требований заказчика.
2. Анализ требований и их проверка.
3. Определение границ системы и интерфейсов.
4. Создание логической и физической модели данных.
5. Разработка архитектуры и определение компонентов системы.
6. Проектирование и описание интерфейсов.
7. Разработка процедур и алгоритмов обработки данных.
8. Написание и отладка кода.
9. Создание модульных и интеграционных тестов.
10. Внедрение необходимых библиотек и инструментов.
11. Верстка пользовательского интерфейса.
12. Разработка тестовых сценариев.
13. Выявление и устранение дефектов.
14. Оценка качества и полноты тестирования.
15. Установка и настройка системы.
16. Обучение пользователей и администраторов.
17. Подготовка документации и инструкций.
18. Проведение приемочных испытаний.
19. Исправление обнаруженных ошибок.
20. Выпуск обновлений и патчей.
21. Поддержка пользователей и решение возникающих проблем.



# Лабораторная работа № 2. Оценка трудозатрат на выполнение работ по разработке и внедрению программного обеспечения

Цели занятия: получить первоначальные навыки применения инженерных подходов к процессам оценки трудоемкости выполнения работ по разработке и внедрению автоматизированных информационных систем, получить первоначальные навыки расчета стоимости выполнения работ по разработке и внедрению автоматизированных информационных систем.

Цель работы: расчет трудоемкости и стоимости выполнения работ по проекту, для которого был составлен календарный план в процессе выполнения лабораторной работы № 1. Расчет трудоемкости следует проводить методов функциональных точек.

## Определите основные процессы, подлежащие автоматизации, постройте соответствующие диаграммы Idef0.

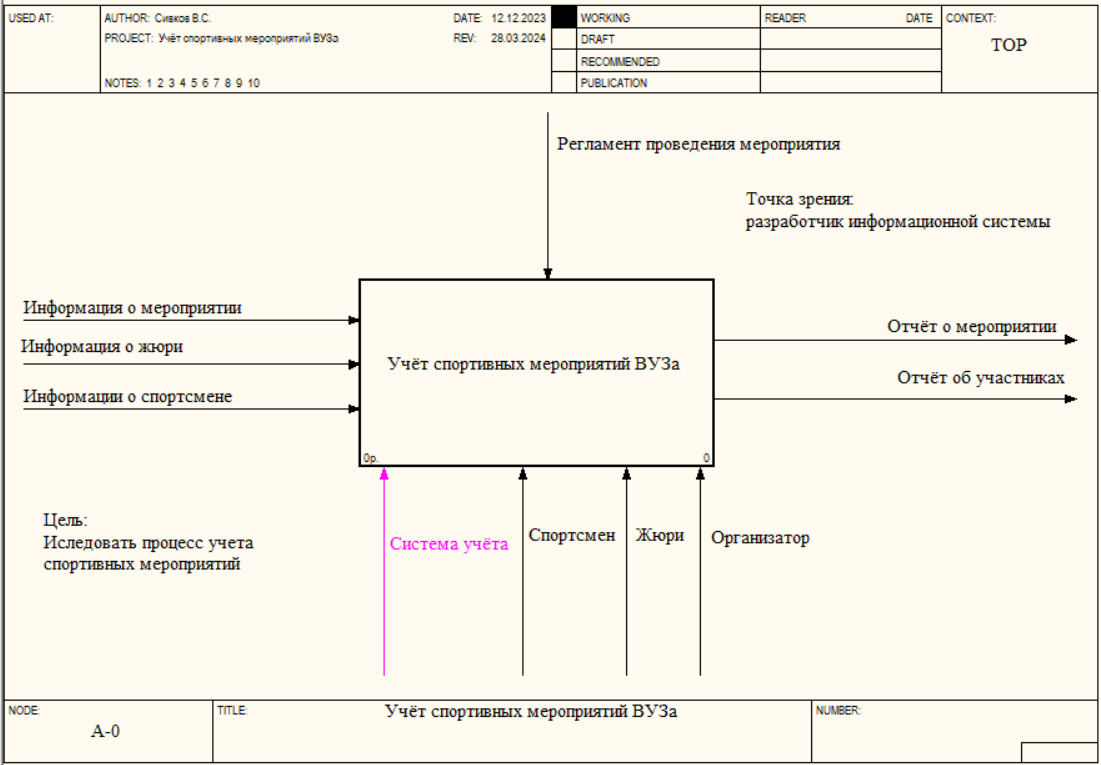


Рисунок . Контекстная диаграмма «TO-BE» IDEF0

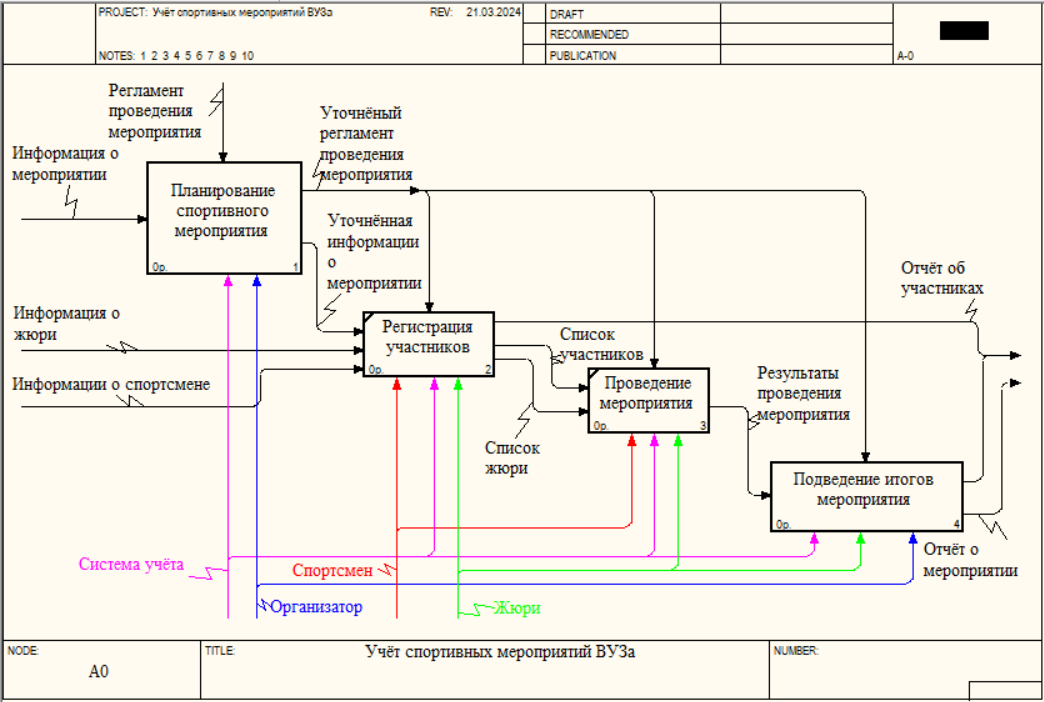


Рисунок . Декомпозиция 1 уровня диаграммы «TO-BE»

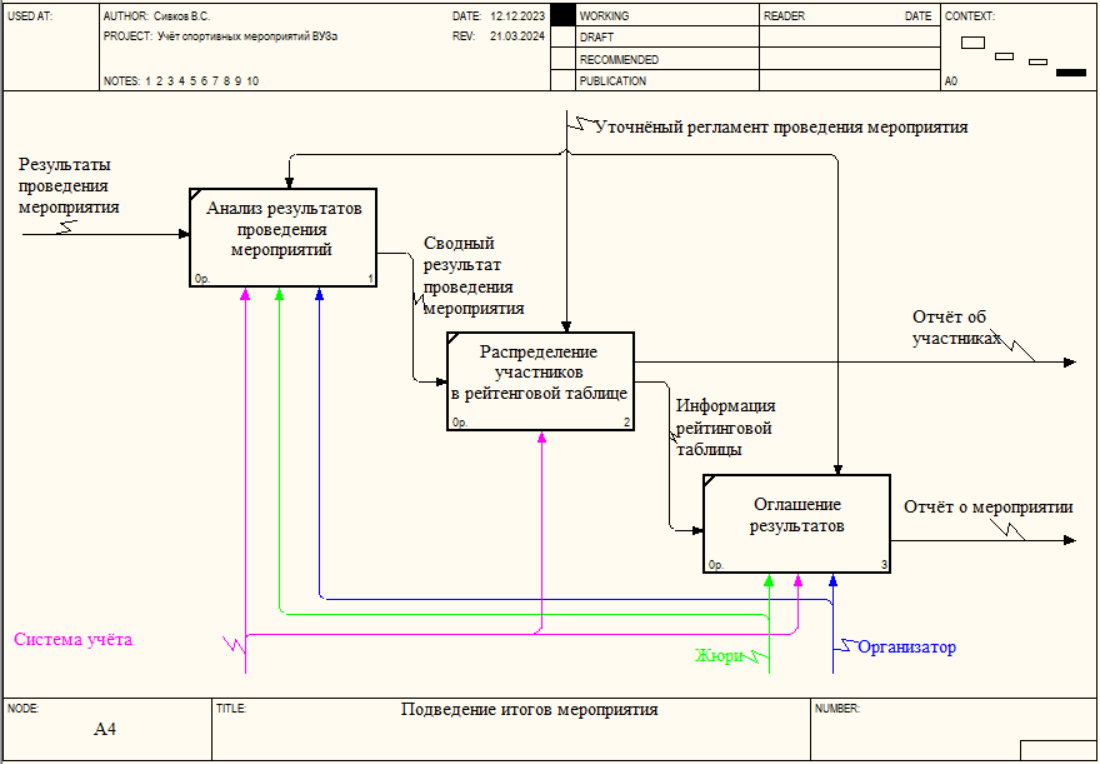


Рисунок . Декомпозиция 2 уровня бизнес-процесса «Подведение итогов мероприятия» в диаграмме «TO-BE»

## Определите основные элементы данных, используемые в автоматизируемых процессах, постройте соответствующие диаграммы Idef1x

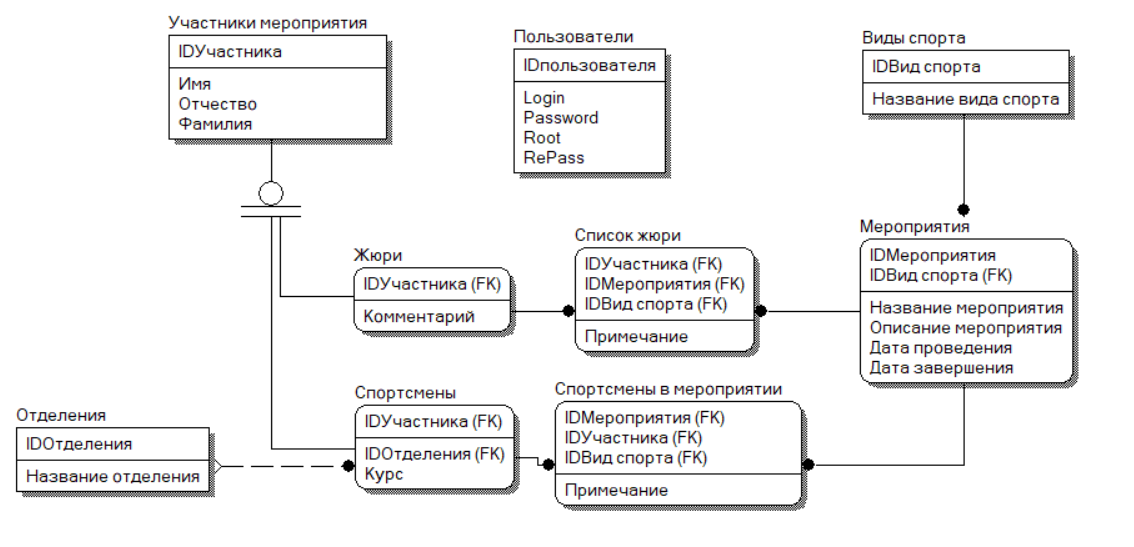


Рисунок . Логическая схема диаграммы IDEF1.x

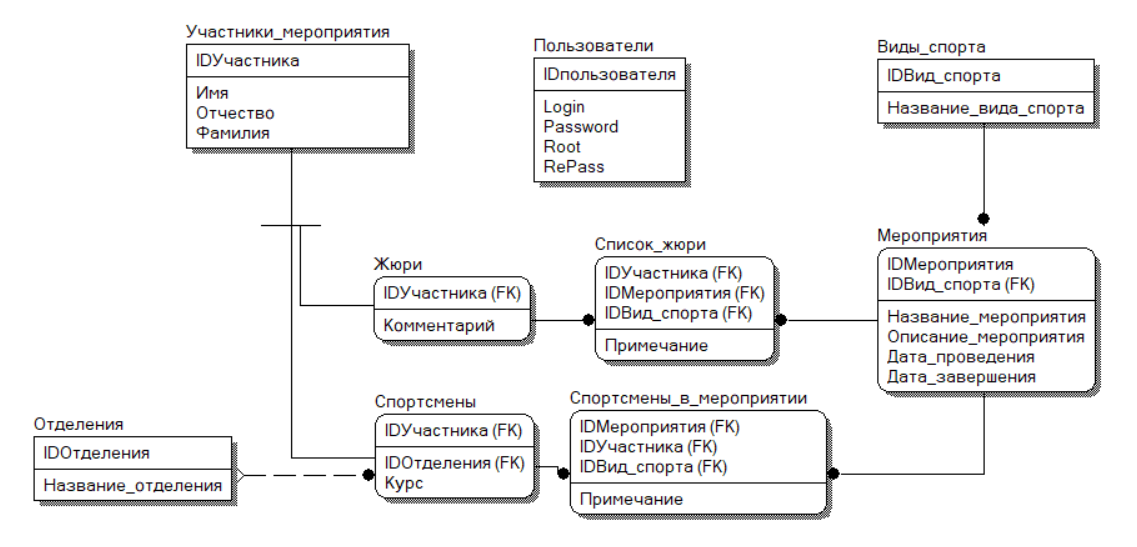


Рисунок . Физическая схема диаграммы IDEF1.x

## Рассчитайте количество функциональных точек для каждого из автоматизируемых процессов.

Таблица . Планирование спортивного мероприятия

| Наименование функции | Количество функциональных  точек, соответствующее  категории функции | | | Количество  функциональных точек |
| --- | --- | --- | --- | --- |
| Простая | Средняя | Сложная |
| 1. Определение   количества выводов | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 2. Определение   количества вводов | 4 | 4 | 5 | 1 |
| **4** | **4** | **5** |
| 3. Определение   количества опросов   вывода | 4 | 4 | 5 | 1 |
| **4** | **4** | **5** |
| 4. Определение   количества опросов   ввода | 4 | 4 | 5 | 1 |
| **4** | **4** | **5** |
| 5. Определение   количества   файлов | 7 | 7 | 7 | 1 |
| **7** | **7** | **7** |
| 6. Определение   количества   интерфейсов | 5 | 5 | 7 | 1 |
| **5** | **5** | **7** |
| Общее количество функциональных точек | | | | 7 |
|  |  |  | FP = | 12,18 |
|  |  |  | LOC = | 53 |

Таблица . Регистрация участников

| Наименование функции | Количество функциональных  точек, соответствующее  категории функции | | | Количество  функциональных точек |
| --- | --- | --- | --- | --- |
| Простая | Средняя | Сложная |
| 1. Определение   количества выводов | 4 | 5 | 7 | 3 |
| **12** | **15** | **21** |
| 2. Определение   количества вводов | 4 | 5 | 7 | 3 |
| **12** | **15** | **21** |
| 3. Определение   количества опросов   вывода | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 4. Определение   количества опросов   ввода | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 5. Определение   количества   файлов | 7 | 10 | 10 | 2 |
| **14** | **20** | **20** |
| 6. Определение   количества   интерфейсов | 7 | 7 | 10 | 3 |
| **21** | **21** | **30** |
| Общее количество функциональных точек | | | | 15 |
|  |  |  | FP = | 52,65 |
|  |  |  | LOC = | 53 |

Таблица . Проведение мероприятия

| Наименование функции | Количество функциональных  точек, соответствующее  категории функции | | | Количество  функциональных точек |
| --- | --- | --- | --- | --- |
| Простая | Средняя | Сложная |
| 1. Определение   количества выводов | 4 | 5 | 7 | 1 |
| **4** | **5** | **7** |
| 2. Определение   количества вводов | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 3. Определение   количества опросов   вывода | 4 | 4 | 5 | 1 |
| **4** | **4** | **5** |
| 4. Определение   количества опросов   ввода | 4 | 4 | 5 | 1 |
| **4** | **4** | **5** |
| 5. Определение   количества   файлов | 7 | 7 | 7 | 0 |
| **0** | **0** | **0** |
| 6. Определение   количества   интерфейсов | 5 | 5 | 7 | 0 |
| **0** | **0** | **0** |
| Общее количество функциональных точек | | | | 5 |
|  |  |  | FP = | 6,95 |
|  |  |  | LOC = | 53 |

Таблица . Подведение итогов мерпориятия

| Наименование функции | Количество функциональных  точек, соответствующее  категории функции | | | Количество  функциональных точек |
| --- | --- | --- | --- | --- |
| Простая | Средняя | Сложная |
| 1. Определение   количества выводов | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 2. Определение   количества вводов | 4 | 4 | 5 | 1 |
| **4** | **4** | **5** |
| 3. Определение   количества опросов   вывода | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 4. Определение   количества опросов   ввода | 4 | 5 | 7 | 2 |
| **8** | **10** | **14** |
| 5. Определение   количества   файлов | 7 | 7 | 7 | 1 |
| **7** | **7** | **7** |
| 6. Определение   количества   интерфейсов | 7 | 7 | 10 | 2 |
| **14** | **14** | **20** |
| Общее количество функциональных точек | | | | 10 |
|  |  |  | FP = | 24,3 |
|  |  |  | LOC = | 53 |

## Составьте смету затрат на выполнение работ по проекту в соответствии с вышеприведенной методикой.

|  |  |  |  |
| --- | --- | --- | --- |
| **Фактические затраты** | **Базовые затраты** | **Оставшиеся затраты** | **Отклонение по стоимости** |
| 0,00 ₽ | 0,00 ₽ | 772 104,00 ₽ | 772 104,00 ₽ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Оставшиеся затраты** | **Фактические затраты** | **Затраты** | **Длительность** |
| Анализ | 15 024,00 ₽ | 0,00 ₽ | 15 024,00 ₽ | 6 дней |
| Проектирование | 132 000,00 ₽ | 0,00 ₽ | 132 000,00 ₽ | 11 дней |
| Разработка | 384 000,00 ₽ | 0,00 ₽ | 384 000,00 ₽ | 32 дней |
| Тетсирование | 18 240,00 ₽ | 0,00 ₽ | 18 240,00 ₽ | 8 дней |
| Внедрение | 222 840,00 ₽ | 0,00 ₽ | 222 840,00 ₽ | 14 дней |

# Лабораторная работа № 3. Автоматизация управления проектом по разработке и внедрению автоматизированной информационной системы

Цели занятия: получение первоначальных навыков использования Issue Tracking System для управления проектами по разработке и внедрению автоматизированных информационных систем.

Цель работы: сформировать в Issue Tracking System план работ, имитируя выполнение отдельных задач, выполнить мониторинг работ, сформировать отчет о ходе выполнения проекта.

Используя сайт «Taiga» (<https://tree.taiga.io/>) для имитирования выполнения отдельных задач спланирована новая разработка «Разработка системы учёта спортивных мероприятий».

Определив задачи и указав сроки исполнения, получилось настроить распределение задач по группам.

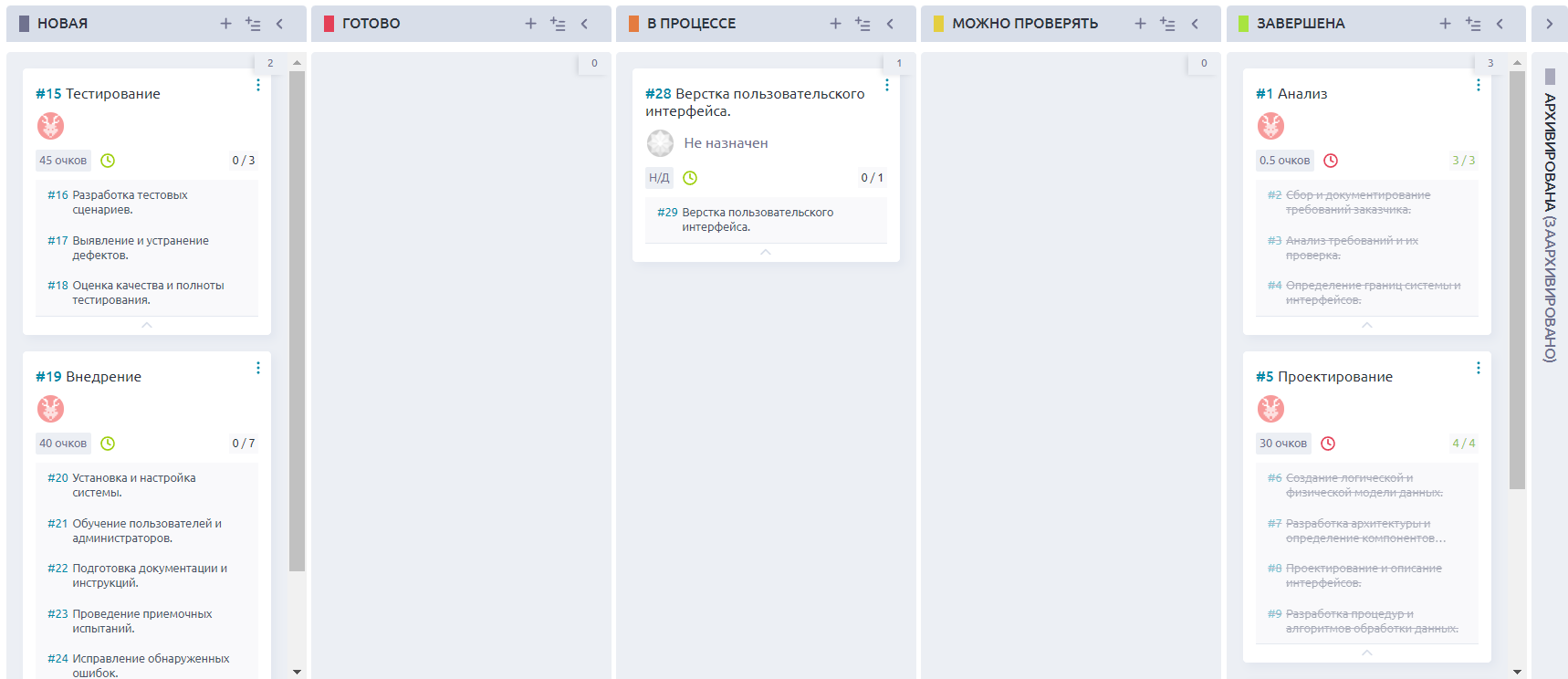


Рисунок .Распланированные задач

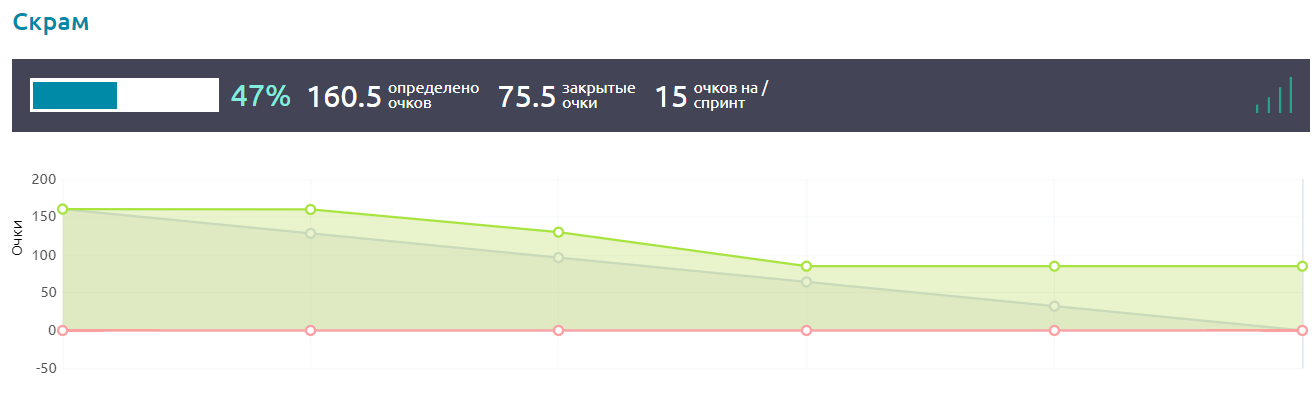


Рисунок . Процент выполнение итоговой работы

Рисунок . Фактические и оставшиеся трудозатраты

# Лабораторная работа № 4. Использование систем контроля версий исходного кода программ.

## Создание нового проекта

GitHub — это облачная платформа для хранения и совместной разработки программного обеспечения. Она использует систему контроля версий Git, которая позволяет нескольким разработчикам работать над одним проектом одновременно. Основные команды Git включают «git init» для создания репозитория, «git add» для добавления файлов, «git commit» для создания коммита, «git push» для отправки изменений на сервер и «git pull» для получения изменений с сервера.

GitHub Desktop — это приложение для работы с GitHub, которое предоставляет графический интерфейс (GUI) для взаимодействия с системой контроля версий Git. Оно помогает разработчикам управлять своими репозиториями, создавать запросы на вытягивание (pull requests), выполнять коммиты и взаимодействовать с коллегами. GitHub Desktop облегчает совместную работу над проектами и соблюдение лучших практик Git.

Основные команды, которые используются для GIT:

git status — показывает состояния файлов в рабочей директории и индексе, какие файлы изменены и ожидают коммита;

git diff — вычисляет разницу между двумя Git деревьями (рабочая директория и индекс, индекс и последний коммит);

git commit — берёт данные из индекса и сохраняет их в базе данных, сдвигает указатель текущей ветки;

git reset — изменяет указатель HEAD и состояние индекса, может изменить файлы в рабочей директории;

git rm — удаляет файлы из индекса и рабочей директории;

git mv — перемещает файл и выполняет git add для нового файла и git rm для старого;

git clean — удаляет мусор из рабочей директории (результаты сборки проекта или файлы конфликтов слияний);

git branch — перечисляет ветки, создаёт новые, удаляет и переименовывает их;

git checkout — переключает ветки и выгружает их содержимое в рабочую директорию;

git merge — сливает одну или несколько веток в текущую ветку, устанавливает указатель текущей ветки на результирующий коммит;

git mergetool — вызывает внешнюю программу для слияния файлов;

git log — показывает историю коммитов, может настроить вывод истории других веток или сравнение между ветками на уровне коммитов;

git stash — временно сохраняет незакоммиченные изменения для очистки рабочей директории без коммита в новую ветку;

git tag — задаёт постоянную метку момента в истории проекта, обычно используется для релизов;

git fetch — связывается с удалённым репозиторием и забирает изменения, сохраняет их локально;

git pull — работает как комбинация команд git fetch и git merge, вначале забирает изменения, затем пытается слить их с текущей веткой;

git push — устанавливает связь с удалённым репозиторием, передаёт локальные изменения в него, требует права на запись;

git remote — управляет списком удалённых репозиториев, сохраняет URL репозиториев в виде понятных строк;

git archive — упаковывает в архив указанные коммиты или весь репозиторий;

git show — отображает объект в простом и человекопонятном виде, обычно используется для просмотра информации о метке или коммите;

git shortlog — подводит итоги команды git log, группирует коммиты по автору;

Используя команду «git init» который создаёт новый репозиторий для хранения версий проекта. Используя учётную запись и приложение «GITHUB desktop» данный репозиторий был загружен на сервер «GITHUB» (см. Рисунок 10).

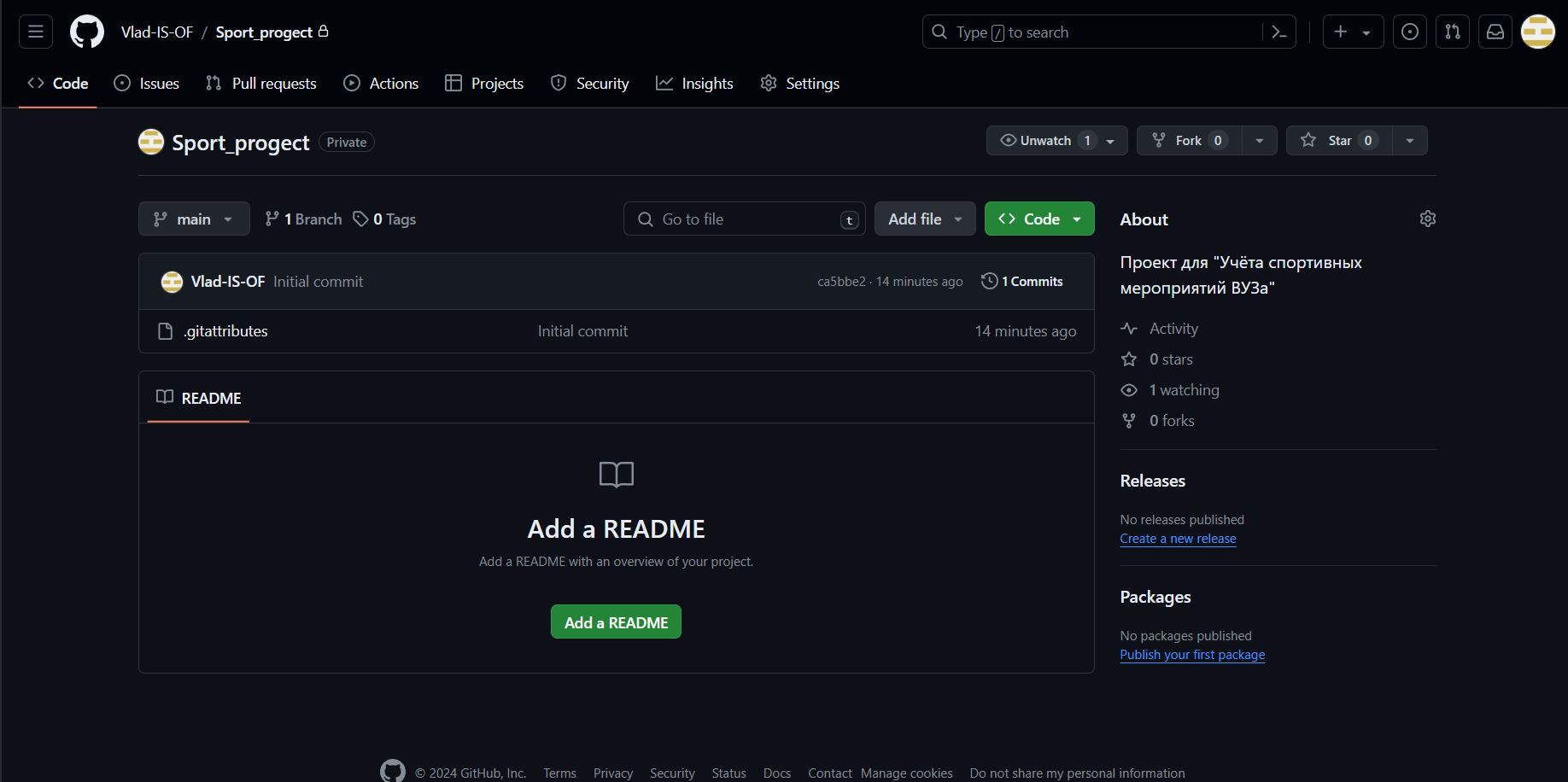


Рисунок . Добавления локального репозитория на сервер «GITHUB»

В созданный репозиторий были загружены файлы проекта. Система сразу предложила произвести первый коммит «git commit» и применить результаты всех изменений (см. Рисунок 11).

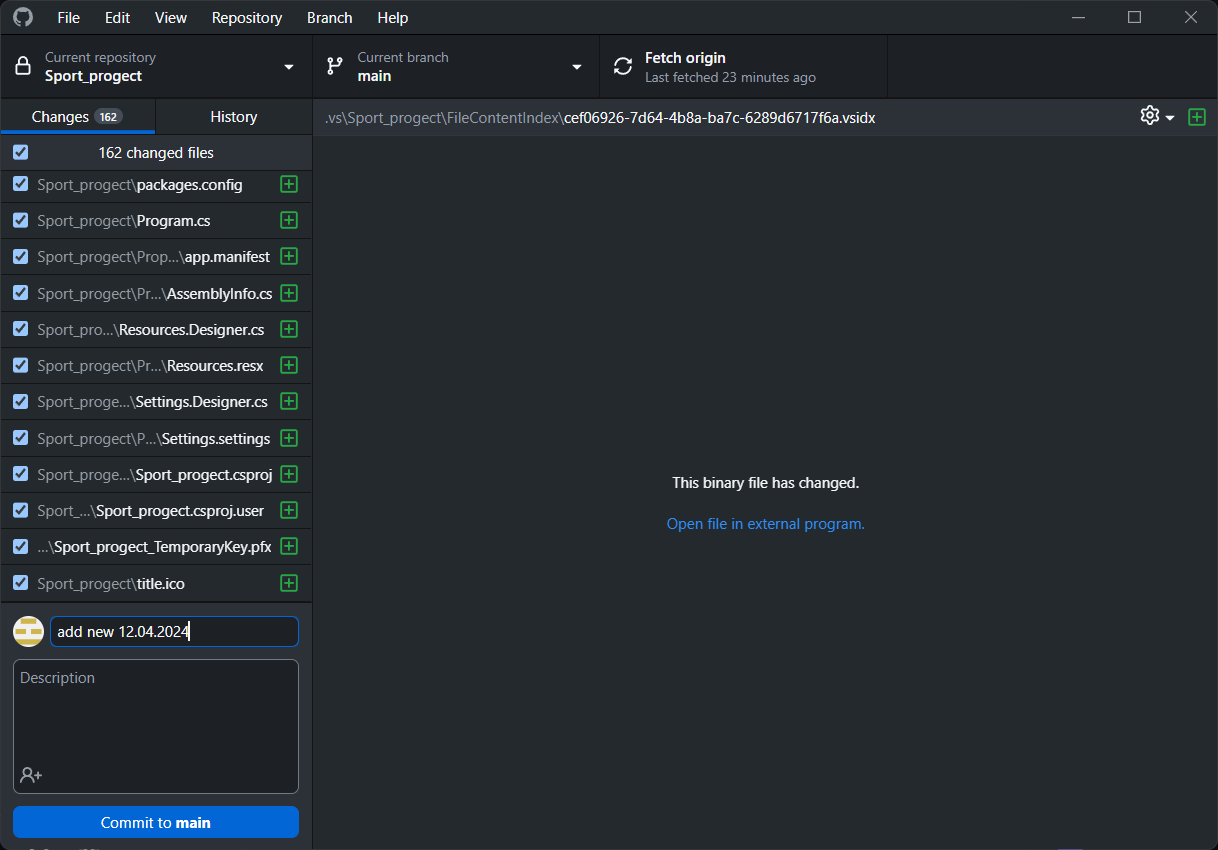


Рисунок . Добавление нового коммита и применение всех изменений в версии.

Изменения были записаны и обновлены в ветки репозиториев можно посмотреть результат и историю изменений (см. Рисунок 12).

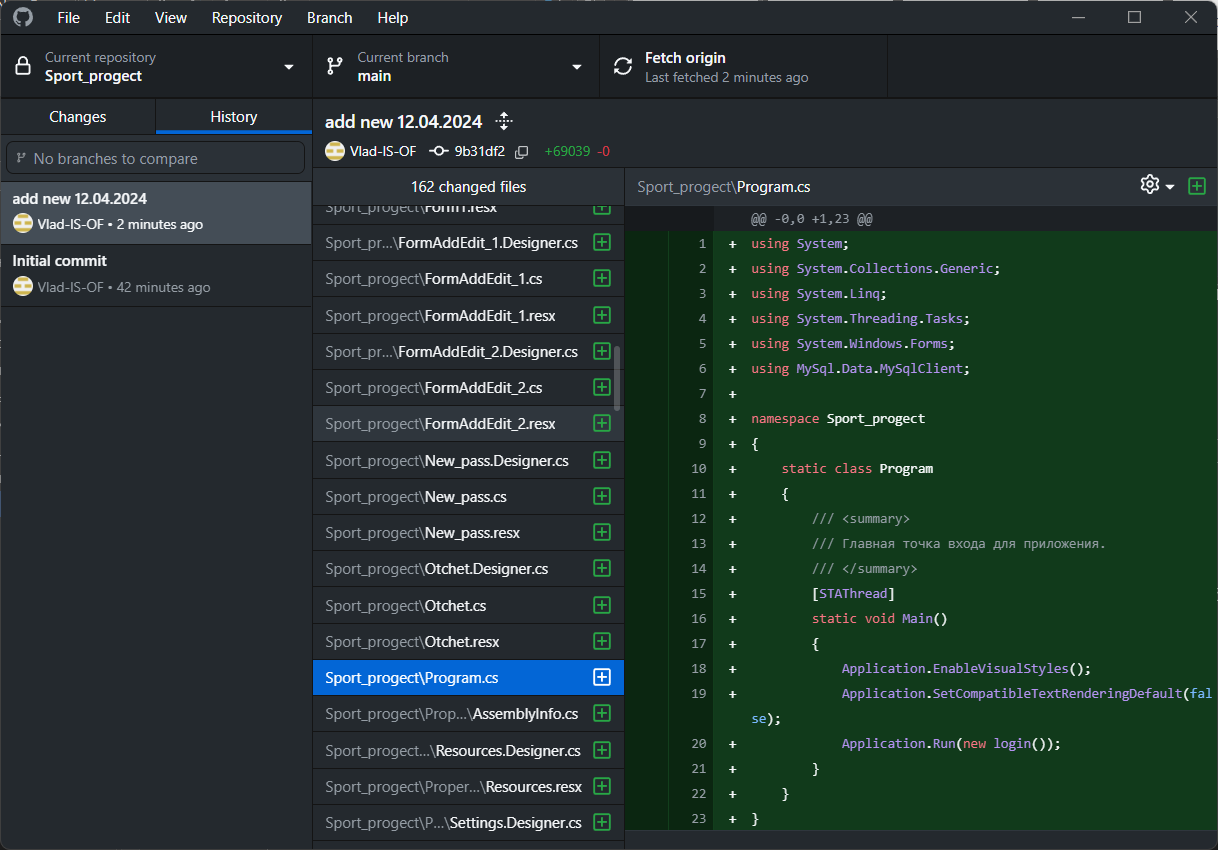


Рисунок . История изменения репозитория

На сервере GITHUB добавил новый текстовый файл и используя команды выгрузить в локальный репозиторий.