

Операционные системы

Отчёт по 1 этапу проекта

Баранов Георгий Павлович

7 марта 2025

Российский университет дружбы народов, Москва, Россия

Цели и задачи

Добавить к сайту данные о себе.

Выполнение лабораторной работы

О себе

👋 Привет! Меня зовут Георгий Баранов. Я студент 1 курса РУДН, учусь на физмате по направ.

Что мне интересно?

- 🧠 Программирование и алгоритмы
- 🧮 Математика и логика
- 💻 Технологии и разработка ПО
- 🤖 Машинное обучение и AI

Что я делаю?

- 📖 Учусь и осваиваю новые технологии
- 🖱️ Разбираюсь с компьютерными науками
- 📝 Пишу о своих открытиях в блоге

Буду рад обсудить интересные темы и обменяться опытом! 😊

Рис. 1: Файл об авторе

📅 Итоги недели

Эта неделя была насыщенной! Вот несколько ключевых событий:

- 🧠 Разобрался с темой ****рекурсивных алгоритмов**** - оказалось проще, чем казалось сначала.
- 🏫 На лабораторной по физике сломали эксперимент, но зато узнали, как не надо делать.
- 📄 Наконец-то понял, как работает ****замыкание**** в программировании. Теперь код выглядит ч
- 📖 Начал читать книгу про ****парадигмы программирования**** - мозг кипит, но это интересно.
- 🌳 Вышел из режима "учеба-дом" и погулял - природа помогает перезагрузиться.

Как прошла твоя неделя? Делись в комментариях! 😊

Рис. 2: Файл для поста

```
# [📄] Управление версиями. Git

## Что такое управление версиями?

Управление версиями – это процесс отслеживания и контроля изменений в коде. Это особенно ва

**Почему это важно**
- ✔ Позволяет откатиться к предыдущей версии кода
- ✔ Помогает отслеживать изменения и их авторов
- ✔ Облегчает совместную разработку

## Git – главный инструмент управления версиями

Git – это самая популярная распределённая система управления версиями (VCS). Она использует

### 📌 Основные принципы работы с Git

1. **Локальный и удалённый репозитории**
  - Git хранит все изменения в **локальном репозитории** на вашем компьютере.
  - Репозиторий можно синхронизировать с **удалённым сервером**, например, на GitHub.

2. **Коммиты и история изменений**
  - **Коммит** (commit) – это сохранение изменений в истории проекта.
  - Каждый коммит имеет уникальный **хеш** (идентификатор), который позволяет легко отслеж

3. **Ветвление и слияние**
  - Git позволяет создавать **ветки** (branches), чтобы работать над новыми фичами без рис
  - Когда работа завершена, ветку можно **объединить** (merge) с основной.

4. **Разрешение конфликтов**
  - Если два разработчика изменили один и тот же файл, может возникнуть **конфликт**.
  - Git позволяет вручную выбрать, какие изменения оставить.

5. **Работа в команде**
  - Разработчики делают изменения в **своих ветках**.
  - После проверки кода происходит **слияние** изменений в основную ветку.
  - Это позволяет избежать хаоса в коде и облегчает контроль качества.
```

Рис. 3: Файл для публикации

Выводы

Добавили к сайту данные о себе.