Санкт-Петербургский государственный университет

Кафедра системного программирования

Улучшение окружения кода miminet для дальнейшего масштабирования

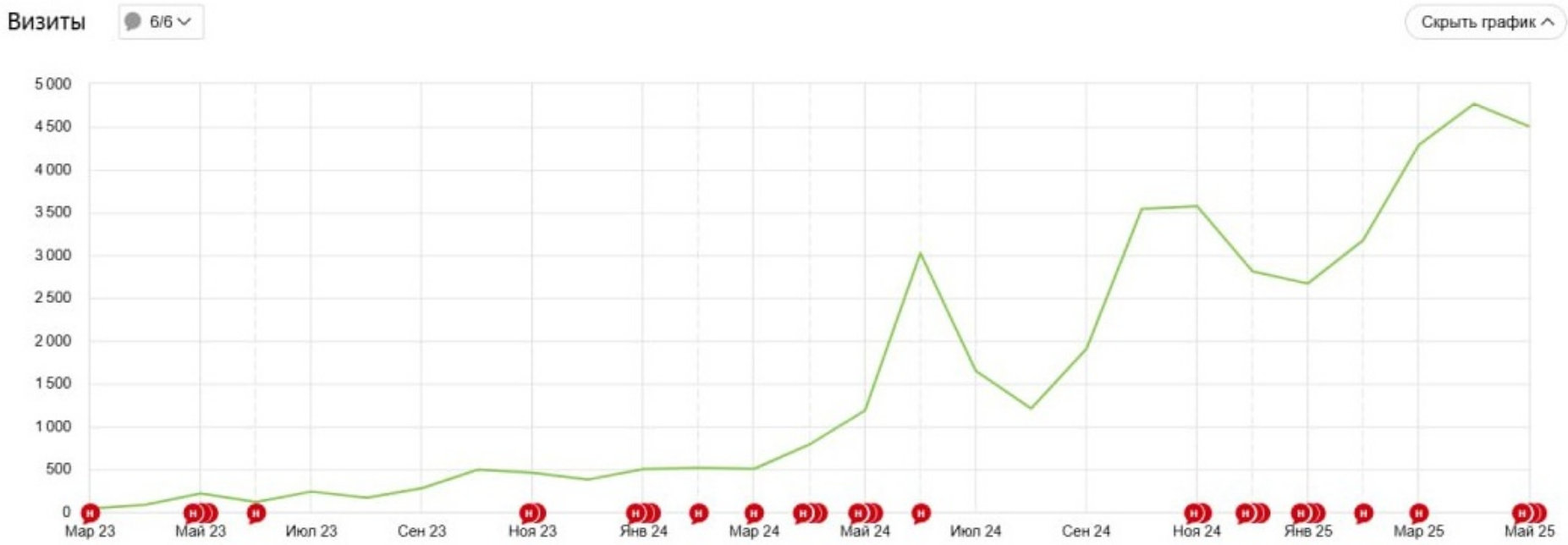
Зайцев Дмитрий Сергеевич, группа 22.Б11-мм

Научный руководитель: старший преподаватель СПБГУ Зеленчук И. В.

Санкт-Петербург
2025

Введение

1. Проект растёт по числу пользователей, количеству хранимых и обрабатываемых данных
2. Появляется необходимость в обеспечении стабильности, рефакторинге кода



Темы выступления

- Результаты работы над системой автоматического тестирования miminet
- Моя помощь в подготовке к YADRO Impulse 2025
- Переход на PostgreSQL
- Рефакторинг бэкенда

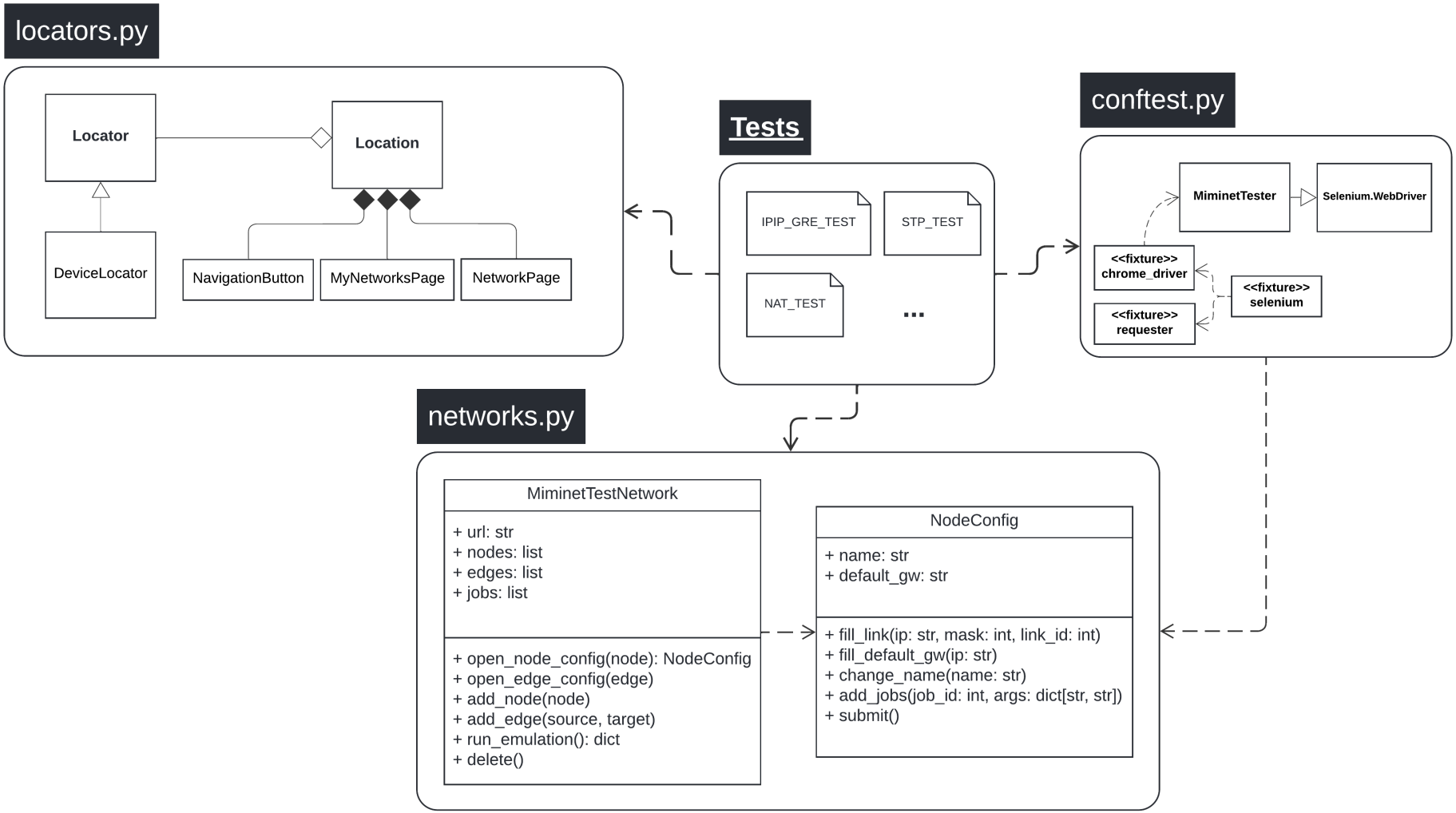
Постановка задачи

Целью работы является создание метода автоматического end-to-end тестирования Miminet.

Задачи:

1. Сбор требований
2. Обзор технологий
3. Создание тестов на ключевые функции Miminet
4. Добавление возможности запуска тестов для разработчиков
5. Интеграция с CI

Итоговая архитектура



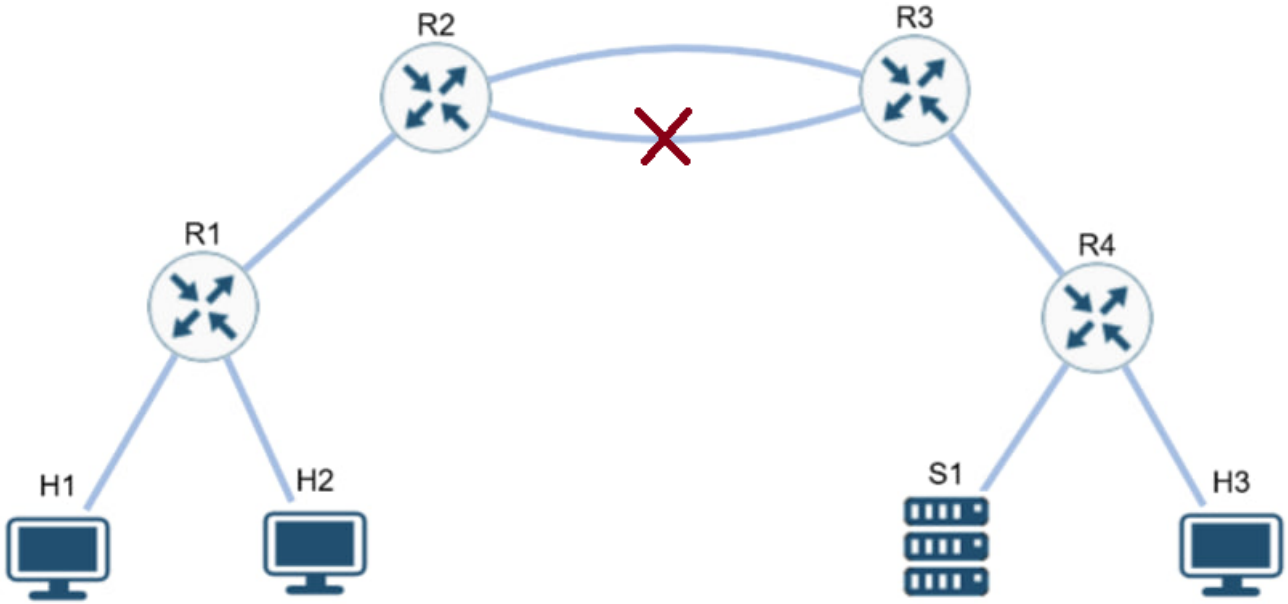
Результаты

1. Выполнен обзор технологий
2. Собраны требования для тестирующей системы
3. Создано 13 тестов на ключевые функции Miminet
4. Реализована возможность запуска тестов для разработчиков
5. Выполнена интеграция с CI
6. Целый семестр успешного использования системы в разработке

Подготовка к Yadro Impulse

Хронология появления проблемы:

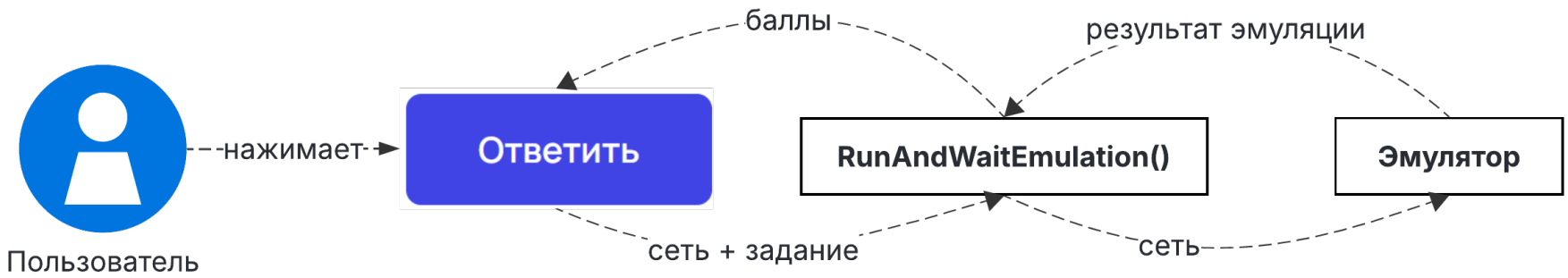
1. За 1–1.5 недели до начала отборочного этапа, поступили условия задач и требования от компании «YADRO»
2. Формулировка одного из условий: «при выходе любого из соединений между R2 и R3 связь не нарушается»



Подготовка к Yadro Impulse

Хронология появления проблемы:

1. За 1–1.5 недели до начала отборочного этапа, поступили условия задач и требования от компании «YADRO»
2. Формулировка одного из условий: «при выходе любого из соединений между R2 и R3 связь не нарушается»
3. Условие нельзя выразить в текущей реализации, система не умеет предобрабатывать сети
4. Задание состоит из множества условий, необходимо проверять каждое со своей модифицированной сетью

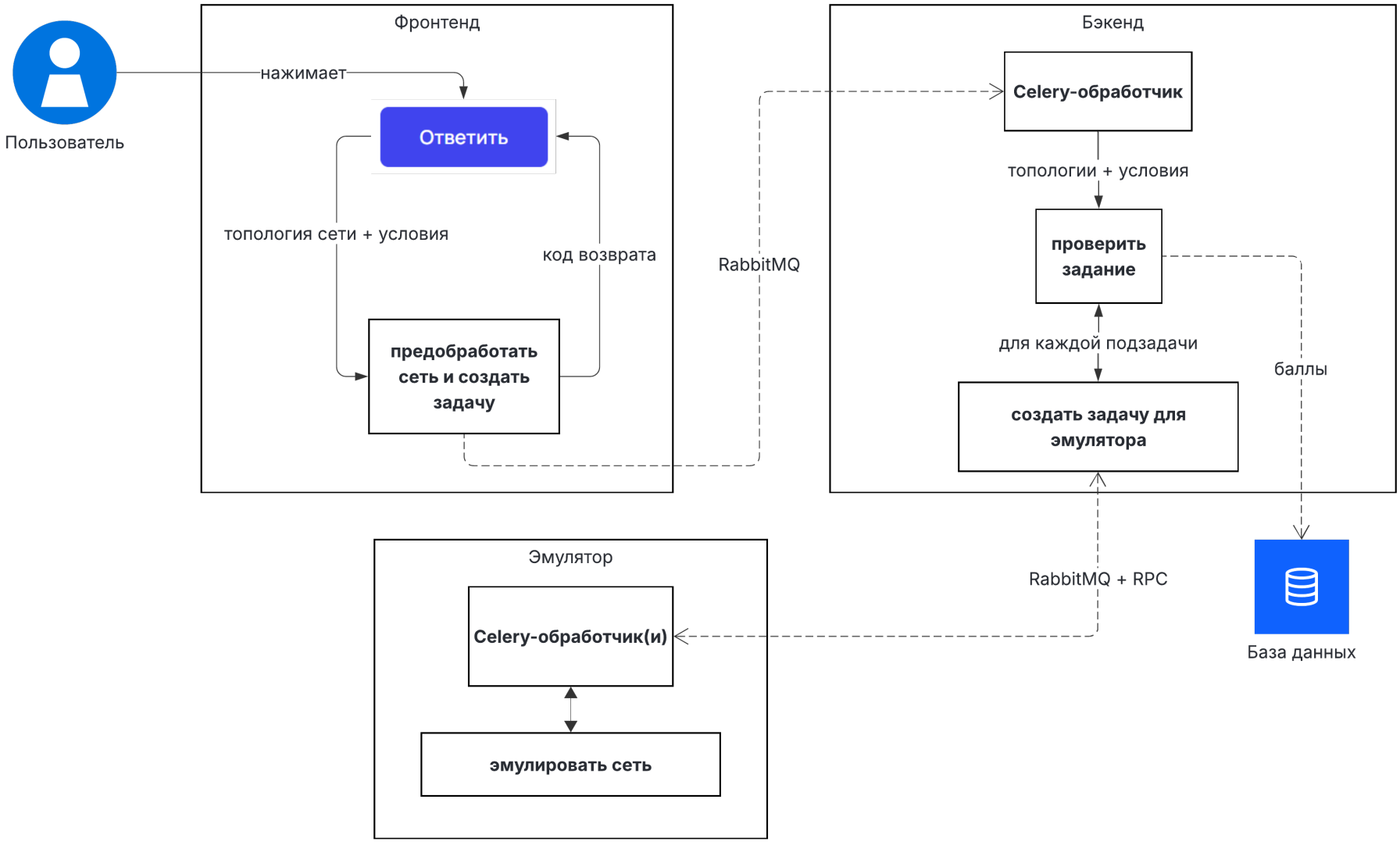


Подготовка к Yadro Impulse

Требования и ограничения при проектировании:

- Нежелательно вносить изменения в эмулятор Miminet
- Обязательно нужен этап *предобработки* пользовательских сетей перед эмуляцией и проверкой
- Проверка одного задания не может быть выполнена за одну эмуляцию
- Пользователь должен отправить задание и узнать свой результат в строго установленную дату
- Простота и читаемость архитектуры

Обновлённая архитектура системы проверки заданий



Переход на PostgreSQL

Предпосылки:

1. Объём БД вырос до $\sim$1 ГБ и продолжает расти по 300 МБ в неделю
2. Начинает заканчиваться место на диске, нужно переносить проект в новое место
3. Запросы в SQLite начинают выполняться медленнее
4. Прежняя реализация доступа к БД была завязана на использовании Volume, подключённого прямо к основному контейнеру приложения
5. Накопились проблемы с запуском, бэкапами, отслеживанием состояния БД

Переход на PostgreSQL

Подготовка:

- Компонент базы данных вынесен в отдельный контейнер Docker
- Новая логика работы с БД внутри приложения
  - Подключение через внутреннюю сеть `docker-compose`
  - Автоматизирована инициализация схемы базы данных
  - Адаптированы типы данных в ORM-модели под PostgreSQL
- Проверка данных в старой БД на конфликты

Переход на PostgreSQL

Этапы перехода:

1. Развёртывание контейнера PostgreSQL на сервере
2. Установка подключения к SQLite и PostgreSQL одновременно из контейнера
3. Работа с некорректными данными
   - Сети у несуществующих пользователей
   - Эмуляции отсутствующих сетей
4. Подготовка конфигурации для `pgloader` с описанием правил миграции
5. Перенос данных из SQLite в PostgreSQL с помощью `pgloader`
6. Тестирование и исправление багов

Рефакторинг бэкенда

Накопившиеся проблемы:

- Сложный и дублирующийся код в эмуляторе
  - Суммарно около 2000 строк кода
  - Изменения вносятся каждые 1-2 недели
- Избыточность и запутанность кода в тестах
  - Разные проверки под разные протоколы, почти полностью идентичные по смыслу
  - Все проверки и файлы с тестами находятся там же, где и основной код
  - Нет читаемых сообщений о месте возникновения ошибки
  - Добавление нового протокола = новые проверки
- Неточности в именовании переменных, классов и файлов
- Вводящие в заблуждение комментарии

Рефакторинг бэкенда

Что было сделано:

- Эмулятор:
  - Основной файл разбит на функции и классы (500 $\rightarrow$ 140 строк в главном файле)
  - Логика конфигурации сетей вынесена в отдельный класс `Mininet`
  - Устранены неточности в именованиях и комментариях
- Тестирование:
  - Проверки сведены в одну параметризованную функцию
  - Дублирующийся код удалён, объём файла с тестами снижен на 50%
  - Тесты вынесены в отдельную директорию `tests/`
  - Добавлено логирование и возможность конфигурации
  - README дополнен инструкцией по запуску
- Общий результат:
  - Повышена читаемость и модульность кода
  - Упростилось добавление новых протоколов и тестов для них

Результаты работы за семестр

1. Создание метода автоматического end-to-end тестирования Miminet
   - выполнен обзор технологий
   - собраны требования для тестирующей системы
   - созданы тесты на основные функции Miminet
   - реализована возможность локального запуска тестов
   - выполнена интеграция с CI
2. Масштабирование и улучшение кода Miminet
   - проект перенесён на новую СУБД PostgreSQL
   - выполнен рефакторинг бэкенда
3. Помощь в подготовке к YADRO ИМПУЛЬС 2025

Репозиторий: `https://github.com/mimi-net/miminet`