

Санкт-Петербургский государственный университет

Кузьмина Ксения Антоновна

Выпускная квалификационная работа

Развитие системы автоматической проверки заданий веб-сервиса Miminet

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2021 «Программная инженерия»*

Научный руководитель:
доцент кафедры системного программирования, к.ф.-м.н. Д. В. Луцев

Консультант:
старший преподаватель И. В. Зеленчук

Рецензент:
стажер отдела разработки ПО для комплаенс-контроля и риск-менеджмента, АО
«Райффайзенбанк» Е. З. Бакова

Санкт-Петербург
2025

Saint Petersburg State University

Kseniia Kuzmina

Bachelor's Thesis

Development of the Automated Assignment Grading System of the Miminet Web Service

Education level: bachelor

Speciality *09.03.04 "Software Engineering"*

Programme *CB.5080.2021 "Software Engineering"*

Scientific supervisor:
C.Sc., docent D. V. Luciv

Consultant:
Senior lecturer I. V. Zelenchuk

Reviewer:
Intern of the Software Development Department for Compliance Control and Risk
Management, JSC "Raiffeisenbank" E. Z. Bakova

Saint Petersburg
2025

Оглавление

Введение	4
1. Постановка задачи	6
2. Обзор	7
2.1. Требования к новой функциональности	7
2.2. Обзор эмуляторов компьютерных сетей	13
3. Архитектура	17
3.1. Используемые технологии	17
3.2. Архитектура веб-сервиса Miminet	18
3.3. Расширение архитектуры подсистемы тестирования . . .	19
3.4. Корректировка архитектуры по итогам апробации	21
4. Особенности реализации	25
4.1. Автоматическая проверка практических задач	25
4.2. Развитие теоретической части подсистемы	29
4.3. Улучшение пользовательского интерфейса	30
5. Апробация	33
Заключение	34
Список литературы	36

Введение

Обучение компьютерным сетям играет ключевую роль в подготовке специалистов по программной инженерии и другим направлениям, связанным с разработкой и эксплуатацией информационных систем. Для эффективного освоения учебного материала необходимо не только изучение теории, но и практическая работа с сетевыми конфигурациями.

Для отработки практических навыков можно использовать реальное сетевое оборудование, однако этот подход связан с рядом трудностей: требуется закупка и обслуживание оборудования, настройка физической инфраструктуры, а также значительные ресурсы времени и пространства. Альтернативный вариант — использование виртуальных машин в среде VirtualBox или VMware — тоже требует от студентов дополнительных знаний в администрировании Linux, а также достаточно мощного компьютера. Частично эти проблемы решаются с помощью инструментов вроде Cisco Packet Tracer или GNS3, однако они имеют ограниченную гибкость и могут быть недостаточно интуитивны для начального уровня обучения.

Для решения этой проблемы могут применяться веб-эмуляторы сетей. Одним из таких инструментов является система Miminet [1] (создана на базе Mininet) [8]. Главное преимущество Miminet заключается в его доступности: преподаватель или студент может открыть браузер и сразу приступить к изучению сетей — без установки Linux и дополнительных утилит.

Помимо основной функциональности по работе с сетями, в Miminet интегрирована подсистема тестирования [19], представляющая собой модуль для проверки знаний пользователей посредством задач различных видов (с выбором варианта ответа, сортировка, сопоставление и практика). Теоретическая часть подсистемы, включающая в себя все виды вопросов, кроме практических, уже была апробирована во время набора на летнюю стажировку Yadro Impulse 2024 [2] году командой Kornfield [3], которая дала обратную связь.

В то же время практическая часть оставалась весьма ограниченной,

включая лишь автоматическую проверку отправки ICMP-пакетов и не охватывая более сложные сценарии работы с сетями. Необходимость расширения этой функциональности стала особенно актуальной после запуска онлайн-курса по компьютерным сетям на платформе Stepik [4], в рамках которого был выделен отдельный раздел с практическими заданиями. Однако отсутствие полноценной автоматической проверки вынуждало обучающихся самостоятельно оценивать правильность выполнения заданий, что снижало объективность и ограничивало возможности использования курса в формализованных образовательных процессах.

Кроме того, команда Kornfield из компании Yadro, принимавшая участие в апробации теоретической части модуля, выразила заинтересованность в использовании практических заданий для отбора стажёров в рамках программы Yadro Impulse. Однако из-за ограничений текущей реализации модуля, включающей только проверку ICMP, полноценное применение такого подхода оказалось невозможным.

В связи с этим была поставлена задача — реализовать и внедрить полноценную систему автоматической проверки практических заданий, охватывающую такие аспекты, как проверка масок подсетей, корректность маршрутизации, маршруты прохождения пакетов и другие параметры, критически важные для понимания принципов работы компьютерных сетей. Реализация данной функциональности позволит значительно повысить образовательную ценность платформы Miminet и расширить сферу её применения.

1. Постановка задачи

Целью данной работы является расширение функциональности подсистемы тестирования по компьютерным сетям в веб-эмуляторе компьютерных сетей Miminet.

Для её достижения были поставлены задачи, перечисленные ниже.

1. Собрать требования к новой функциональности.
2. Провести обзор существующих решений.
3. Модернизировать архитектуру подсистемы тестирования.
4. Разработать новую функциональность.
5. Провести апробацию.

2. Обзор

Обзор проводится для анализа текущего состояния системы Miminet, выявления её ограничений, а также для изучения существующих решений в области эмуляторов компьютерных сетей и систем автоматической проверки заданий. Это позволяет определить направления для улучшения функциональности и выбрать наиболее подходящие методы реализации.

2.1. Требования к новой функциональности

Требования к новой функциональности были сформулированы на основе обратной связи от преподавателей Санкт-Петербургского государственного университета, студентов курса «Основы компьютерных сетей» на платформе Stepik и команды Kornfield компании Yadro.

2.1.1. Требования к практической части подсистемы

Необходимость модернизации практической части подсистемы тестирования Miminet была особенно актуальна для команды Yadro, которая использовала её в 2024 году для отбора кандидатов на стажировку. Хотя теоретический тест проводился автоматически, проверка практических навыков настройки сетей в Miminet требовала личного вмешательства специалистов компании. Это значительно увеличивало временные затраты на оценку кандидатов и снижало эффективность отборочного процесса.

Исходное состояние практической части подсистемы Miminet можно охарактеризовать как крайне ограниченное и находящееся на уровне прототипа. Функциональность автоматической проверки заданий сводилась лишь к базовой проверке факта отправки ICMP-пакета между двумя хостами, без учёта корректности сетевой конфигурации. Это создавало следующие проблемы: неполноценная проверка сетевых решений и бинарная система оценивания.

В рамках первой проблемы подсистема не учитывала топологию

сети, позволяя кандидатам получать положительный результат при некорректной настройке (например, при прямом соединении хостов, минуя данный в условии маршрутизатор, как показано на рисунках 1 и 2). Однако для отбора кандидатов было важно, чтобы подсистема могла автоматически проверять корректность всей сетевой конфигурации, а не просто факт прохождения пакета.

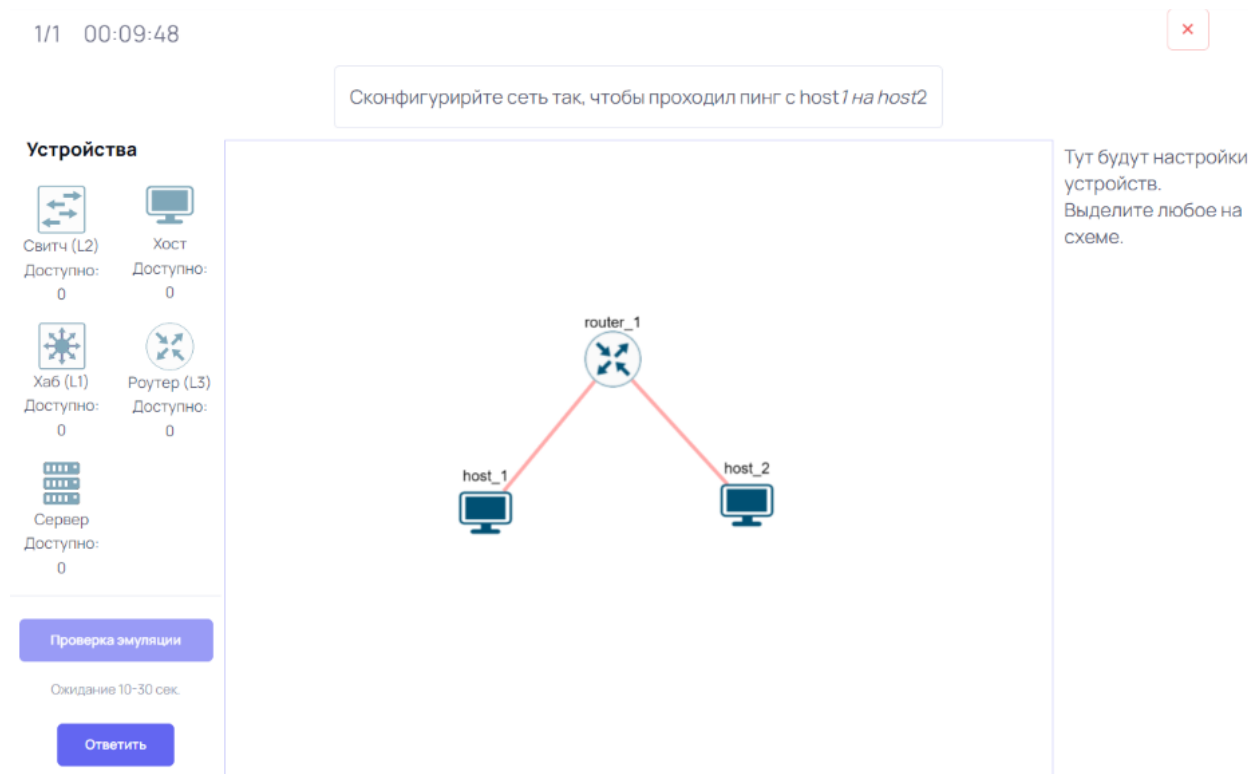


Рис. 1: Стартовая конфигурация с условием задания.

Сконфигурируйте сеть так, чтобы проходил пинг с host 1 на host 2

Устройства

Свитч (L2)

Доступно: 0

Хост

Доступно: 0

Хаб (L1)

Доступно: 0

Роутер (L3)

Доступно: 0

Сервер

Доступно: 0

Верно! Посмотреть результаты

```

graph TD
    router_1((router_1)) --- host_1[host_1  
192.168.0.1/24  
(ping -c 1 192.168.0.2)]
    router_1 --- host_2[host_2  
192.168.0.2/24]
    host_1 --- host_2
  
```

Имя хоста
host_2

Выполнить команду
--

Линк к
router_1

IP-адрес / Маска
/

Линк к
host_1

IP-адрес / Маска
192.168.0.2 / 24

Шлюз по умолчанию
/

Сохранить

Рис. 2: Верное решение задания без учёта роутера, данного в стартовой конфигурации.

Более того, отсутствовала проверка таких параметров, как настройка VLAN, IP-адресации, таблиц маршрутизации и других критических аспектов, что не позволяло полноценно оценить навыки кандидатов.

Бинарная система оценивания приводила к тому, что результат проверки был либо положительным, либо отрицательным, без возможности частичного зачёта задания. Для объективной оценки кандидатов необходимо было внедрить дифференцированную систему баллов.

2.1.2. Требования к теоретической части подсистемы

Существующая реализация теоретической части подсистемы тестирования имела ряд существенных ограничений, затрудняющих её использование в образовательном процессе, в частности, для проведения зачёта по курсу компьютерных сетей.

1. Невозможно было вставлять изображения, код или формулы в текст вопросов.
2. Вопросы были жестко привязаны к конкретным тестам и разделам, что позволяло студентам просто заучить их. Кроме того, от-

существовал механизм случайной выборки вопросов из банка по темам.

2.1.3. Требования к разработке API для создания заданий

Существующий интерфейс администратора для создания заданий на базе Flask Admin имел серьезные ограничения в работе с банком вопросов, что создавало значительные трудности при создании и редактировании тестовых материалов. Основные проблемы реализации перечислены ниже.

1. Жёсткая иерархическая структура:

- подсистема требовала строгой последовательности создания элементов: сначала тест, затем раздел теста, потом вопрос и только затем ответ;
- невозможно было создать автономный вопрос для банка вопросов без привязки к конкретному тесту и разделу;
- отсутствие возможности массового создания и импорта вопросов.

Miminet Admin Home Test Section Question Answer

List Create

Текст вопроса

На каком уровне модели ISO/OSI работает IP-протокол?

Пояснение

Вопрос раздела

Not a valid choice

Тип вопроса

С вариантами ответов

Save Save and Add Another Save and Continue Editing Cancel

Рис. 3: Подсистема не даёт создать вопрос, если он не прикреплён к разделу.

2. Неудобный интерфейс:

- каждый элемент (тест, раздел, вопрос, ответ) требовал отдельного экрана редактирования; вкладки можно увидеть на рисунке 3;
- процесс создания даже одного вопроса занимал 4-5 шагов;
- невозможно было быстро скопировать или дублировать существующие вопросы.

Поэтому для более удобного создания вопросов, преподавателем курса "компьютерные сети", имеющим доступ к созданию вопросов, были сформулированы следующие требования к API.

1. Возможность создания вопросов независимо от тестов и разделов.
2. Система тегов для категоризации вопросов.
3. Поддержка различных типов вопросов (с вариантами ответов, на сортировку, сопоставление терминов, практические).

4. Возможность прикрепления медиа-вложений.
5. JSON-based API с четкой схемой валидации.
6. Совместимость с существующей системой аутентификации пользователя.

Такая реализация значительно упростит процесс наполнения подсистемы тестовыми материалами и позволит администраторам сосредоточиться на содержательной части, а не на преодолении ограничений интерфейса.

2.1.4. Требования к пользовательскому интерфейсу

На основе обратной связи от пользователей системы Miminet, включая студентов, преподавателей и представителей компании Yadro, были сформулированы следующие требования к пользовательскому интерфейсу подсистемы автоматической проверки заданий.

1. Детализированный вывод результатов.
 - Для случаев использования подсистемы в рамках отбора на стажировку она должна предоставлять отдельную страницу с подробным выводом результата выполнения заданий, вместо количества верных ответов за весь тест, как показано на рисунке 4. Она должна отображать информацию о допущенных ошибках и набранных баллах за практическое задание, когда появится функциональная возможность ставить за них дифференцированную оценку, чтобы работодатели могли оценить не только общий результат, но и конкретные компетенции кандидата.

Основы компьютерных сетей Практические задачи

Правильных ответов: 1 из 1

Тест пройден за: 0:08:54

[Назад к тестам](#)

Рис. 4: Интерфейс вывода результатов без подробной информации об ошибках пользователя.

- Поддержка мультимедийного контента.
 - (а) В теоретических вопросах должна быть реализована возможность вставки изображений, фрагментов кода и формул. Это особенно важно для курсов по компьютерным сетям, где визуализация топологий сетей или примеры конфигураций могут значительно упростить понимание материала.

Эти требования направлены на повышение удобства использования подсистемы как для обучающихся, так и для преподавателей и работодателей, что в итоге способствует более эффективному обучению и объективной оценке навыков.

2.2. Обзор эмуляторов компьютерных сетей

Теоретическая часть подсистемы тестирования уже была разработана предыдущими участниками проекта и успешно используется в образовательном процессе, несмотря на необходимость некоторой доработки. Однако практическая часть подсистемы находилась в зачаточном состоянии – по сути, представляя собой лишь заглушку с базовой проверкой ICMP-пакетов. При этом именно практические задания вызывали наибольший интерес как у преподавателей СПбГУ, так и у команды Kornfield.

Поскольку предстояло разрабатывать систему автоматической проверки практических заданий практически с нуля, был проведен анализ

существующих решений в области сетевых эмуляторов и систем автоматизированной проверки заданий:

1. **Cisco Packet Tracer** [13] – проприетарный сетевой симулятор от Cisco, широко используемый в образовательных программах компании. У данного программного обеспечения можно выделить следующие характеристики:

- реалистичная эмуляция оборудования Cisco;
- встроенная система проверки заданий через Activity Wizard;
- возможность создания пошаговых инструкций для обучающихся;
- ограниченная поддержка не-Cisco оборудования;
- отсутствие веб-интерфейса и API для интеграции.

2. **Boson NetSim** [20] – коммерческий продукт для эмуляции сетевых конфигураций. NetSim имеет следующие ключевые характеристики:

- подробные пошаговые лабораторные работы с системой подсказок;
- встроенные сценарии проверки конфигураций;
- хорошая документация и обучающие материалы;
- закрытая архитектура без возможности интеграции;
- высокая стоимость лицензирования.

Особый интерес представляет система инструкций NetSim, которая помогает обучающимся при выполнении лабораторных работ – эта функциональность оказалось достаточно интересной, и благодаря ей появилась идея добавить в Miminet не только контрольный, но и учебный режим, чтобы в случае неверного выполнения задания, пользователь мог узнать, в чём он ошибся.

3. **GNS3** [14] – открытая платформа для эмуляции сложных сетевых инфраструктур. У GNS3 можно выделить следующие отличительные черты:

- поддержка виртуализации сетевых устройств;
- возможность интеграции с реальным оборудованием;
- развитое API и сообщество разработчиков;
- высокие требования к ресурсам;
- отсутствие встроенной системы проверки заданий.

Проведенный анализ показал, что ни одно из существующих решений не удовлетворяет требованиям проекта полностью: коммерческие продукты (Cisco Packet Tracer, Boson NetSim) имеют закрытую архитектуру и не поддерживают интеграцию с веб-платформами, а открытые решения (GNS3) не предоставляют готовых механизмов автоматической проверки заданий, к тому же потребляют достаточно много ресурсов. При этом отдельные функциональные элементы, такие как система подсказок из Boson NetSim, представляли ценность для образовательного процесса.

На основе этого анализа было принято решение продолжить развитие собственного решения Miminet, сосредоточившись на следующих аспектах:

- реализация гибкой системы проверки практических заданий с поддержкой различных сетевых сценариев;
- разработка учебного режима с контекстными подсказками по аналогии с Boson NetSim;
- обеспечение веб-доступа и интеграции с существующей платформой;
- создание дифференцированной системы оценивания.

Такой подход позволяет сохранить преимущества существующего решения (веб-доступ, наличие теоретических вопросов) при этом значительно расширив функциональность в области практических заданий.

3. Архитектура

3.1. Используемые технологии

Miminet разрабатывается уже несколько лет [5], поэтому все используемые технологии были выбраны до начала данной работы.

- Flask [16] – популярный фреймворк для разработки веб-приложений и API на языке Python. Его преимущества включают лёгкость освоения, поддержку шаблонов и гибкость в создании серверной логики. В данном проекте Flask используется для обработки HTTP-запросов и взаимодействия с базой данных.
- Jinja2 [17] – встроенный в Flask шаблонизатор, позволяющий динамически генерировать HTML-страницы. Использование Jinja2 обеспечивает удобную интеграцию Python-кода в шаблоны, что упрощает создание и поддержку интерфейса приложения.
- JavaScript [11] – язык программирования, применяемый для добавления интерактивности на веб-страницах. В проекте используется совместно с AJAX для динамической загрузки данных без необходимости полной перезагрузки страницы, что улучшает удобство и скорость работы приложения.
- jQuery [18] – библиотека JavaScript, которая облегчает работу с элементами DOM, управление событиями и обработку запросов AJAX. Её использование также помогает обеспечить кроссбраузерную совместимость и упрощает клиентскую часть приложения.
- Bootstrap [15] – библиотека для создания адаптивных пользовательских интерфейсов. Она предоставляет готовые компоненты и стили, ускоряя разработку страниц с современным и единообразным дизайном.
- SQLite [12] – встроенная реляционная база данных, которая используется для хранения и управления данными проекта. Её лёг-

кость и простота конфигурации идеально подходят для текущих задач приложения.

- SQLAlchemy [9] – ORM-библиотека для Python, упрощающая взаимодействие с реляционными базами данных. В проекте она обеспечивает удобную работу с данными, предоставляя инструменты для построения запросов и управления объектами.
- Docker [10] – платформа для контейнеризации приложений. Позволяет изолировать отдельные компоненты системы в независимых контейнерах, обеспечивая удобство развёртывания, масштабируемость и воспроизводимость окружения.

3.2. Архитектура веб-сервиса Miminet

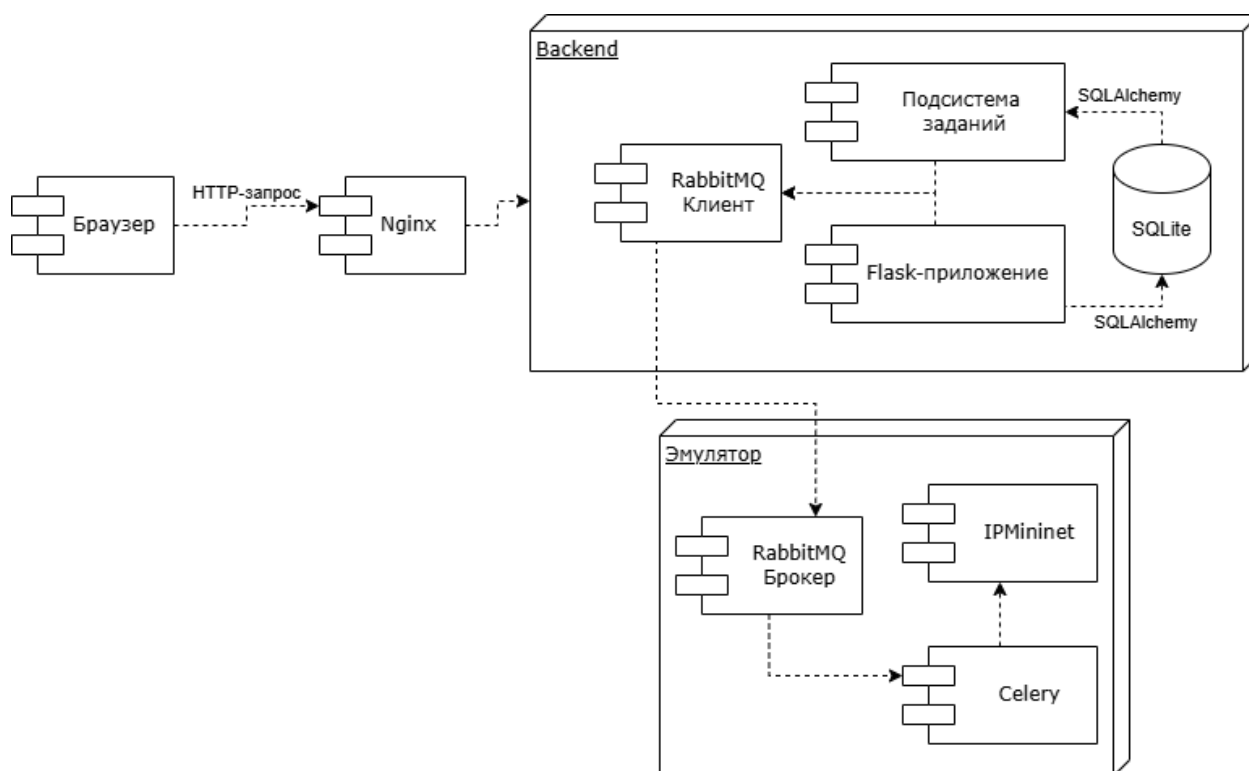


Рис. 5: Диаграмма компонентов веб-сервиса Miminet.

На рисунке 5 представлена диаграмма компонентов веб-сервиса Miminet. Пользователь взаимодействует с приложением через веб-браузер, который отправляет запросы на сервер Nginx, перенаправляющий их либо

на Flask-приложение, либо в подсистему тестирования, запущенные в контейнере.

Flask-приложение реализует серверную логику системы, включая маршрутизацию запросов, обработку данных и взаимодействие с базой данных. Подсистема тестирования отвечает за все этапы работы пользователя во вкладке "Тестирование". Взаимодействие с базой данных осуществляется при помощи ORM-библиотеки SQLAlchemy и встроенной реляционной базы SQLite.

Для выполнения задач, связанных с эмуляцией сетей, используется отдельный контейнер `back-miminet`, где работает Celery. Обмен сообщениями между модулями организован при помощи брокера RabbitMQ, запущенного в контейнере `rabbitmq`. Flask-приложение формирует задания и отправляет их в очередь, откуда они извлекаются Celery и передаются в библиотеку IPMininet, осуществляющую эмуляцию сетевых конфигураций.

Таким образом, архитектура сервиса разделена на логически независимые компоненты: веб-интерфейс, серверная логика, подсистема тестирования, брокер сообщений и система эмуляции. Такой подход обеспечивает расширяемость и упрощает внедрение новой функциональности.

3.3. Расширение архитектуры подсистемы тестирования

На начальном этапе разработки проверка практических заданий в системе была реализована в виде одного монолитного модуля, выполнявшего исключительно проверку прохождения ICMP-пакета (ping) между двумя хостами. На вход модулю подавалась сэмулированная пользователем конфигурация сети в формате JSON, а также JSON-описание условий задания, сформулированное преподавателем. Модуль анализировал переданные пакеты, перебирал их и определял, был ли ICMP-запрос отправлен от нужного хоста и получен указанным адресатом. При соблюдении всех условий возвращалась бинарная оценка:

выполнено или не выполнено.

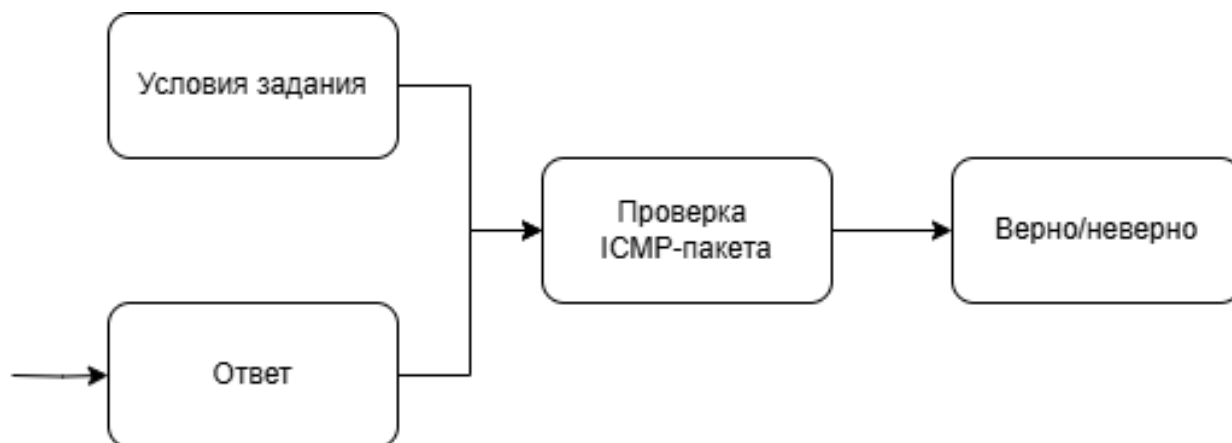


Рис. 6: Схематичное изображение проверки практического задания до начала работы

Примером условия задания в прежней версии подсистемы служит следующий JSON-словарь:

```
{ 'task': 'ping 1 host', 'from': 'host_1', 'to': 'host_2' }
```

В ходе проектирования обновлённой архитектуры подсистемы тестирования было принято решение отказаться от статической и негибкой схемы проверки в пользу модульного и расширяемого подхода. Основной идеей переработки стало введение возможности задавать произвольное количество независимых условий, каждое из которых представляет собой атомарную проверку заданной конфигурации. Для их описания был разработан единый JSON-формат, в котором каждое условие содержит параметры проверки и вес в баллах, позволяющий гибко управлять значимостью различных требований.

Разработанные проверки были условно классифицированы по типу устройства: хосты, маршрутизаторы, серверы, а также общие параметры всей сети (например, настройка VLAN, использование частных IP-адресов, наличие туннелей и т.д.). Для каждой группы условий реализован отдельный модуль, отвечающий за проверку конкретного типа требований. В случае успешного выполнения условия модуль возвращает соответствующее количество баллов, заданное в конфигурации.

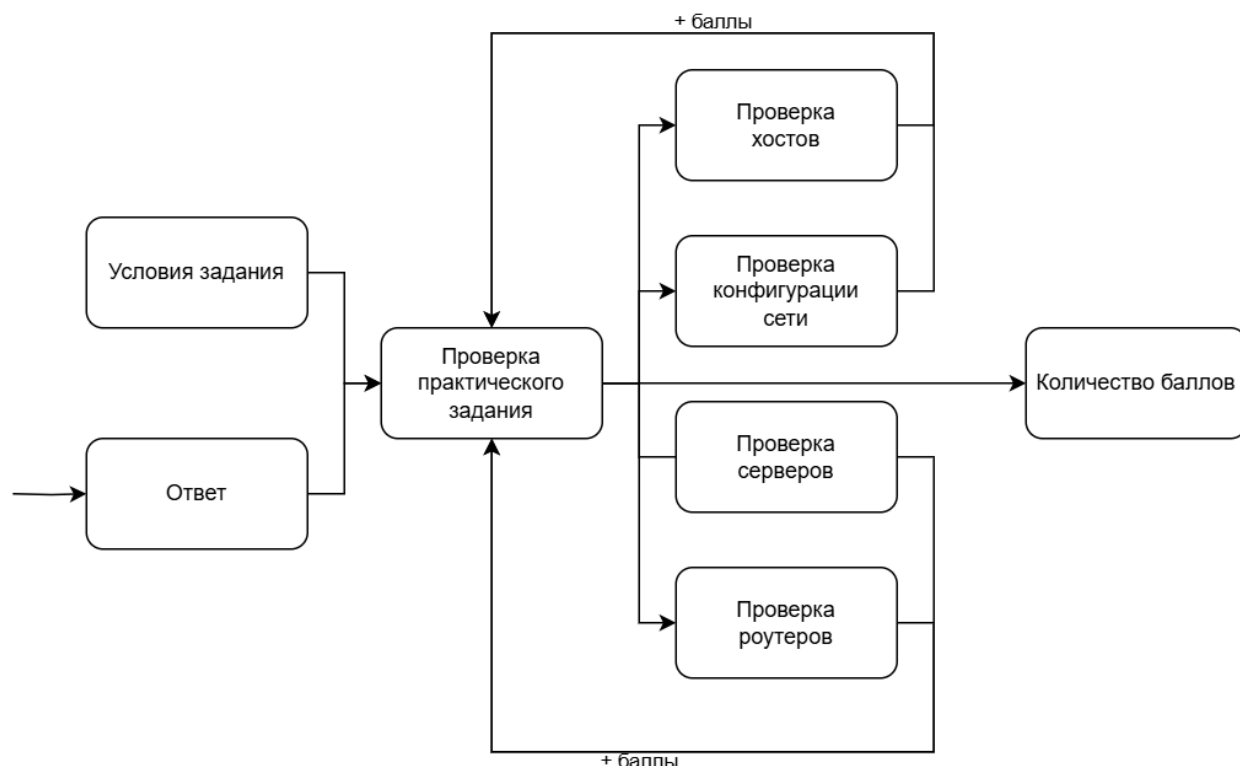


Рис. 7: Схематичное изображение проверки практического задания после переработки архитектуры

Предложенный подход позволил существенно упростить процесс добавления новых видов проверок и расширить возможности подсистемы без необходимости внесения изменений в основную логику. Кроме того, новая архитектура обеспечивает более точную и объективную оценку практических заданий, включая случаи с комплексной маршрутизацией и нетривиальными сценариями настройки. Благодаря гибкой настройке баллов преподаватель или заказчик может самостоятельно определить приоритетность различных условий.

3.4. Корректировка архитектуры по итогам апробации

После завершения разработки базовой архитектуры подсистемы тестирования и реализации её первой версии, она была передана команде Kornfield на предварительную апробацию перед отбором кандидатов на летнюю стажировку Yadro Impulse. По результатам тестирования выяснилось, что реализованная архитектура не в полной мере соответствует

предъявленным требованиям.

Подсистема тестирования изначально была спроектирована таким образом, что в эмуляцию передавалась целиком пользовательская конфигурация, включая вручную настроенные отправки ICMP-пакетов (ping). Таким образом, проверка производилась исключительно на основе тех действий, которые пользователь явно задал в интерфейсе.

Однако, для задач, предлагаемых Kornfield в рамках отбора кандидатов на стажировку, требовалась возможность проверять противоречащие друг другу условия в рамках одной конфигурации. В частности, как показано на рисунке 8, необходимо было убедиться, что пинг между двумя хостами проходит разными путями в разные стороны (по верхнему и нижнему ребру), а также его прохождение сохраняется в случае удаления одного из соединений. При этом пользователь не может вручную настроить альтернативные сценарии внутри одного задания – подсистема на момент апробации не поддерживала проверку множественных сетевых модификаций.

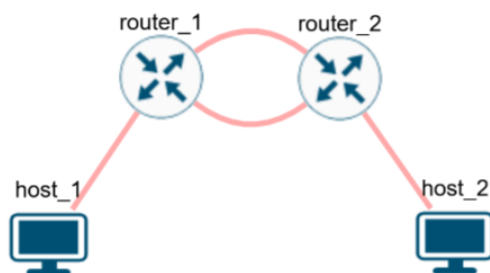


Рис. 8: Упрощённая версия задачи, присланной командой Kornfield

Изначальная архитектура подразумевала работу через две основные пользовательские команды – «Эмулировать» и «Ответить». Обе вызывали одну и ту же функцию симуляции `RunAndWaitSimulation`, однако с разным поведением: в режиме эмуляции пользователь мог просматривать результат и при необходимости внести изменения в конфигурацию; в режиме отправки ответа происходила проверка и блокировка изменений, а потом количество баллов возвращалось в пользовательский интерфейс, как показано на рисунке 9.

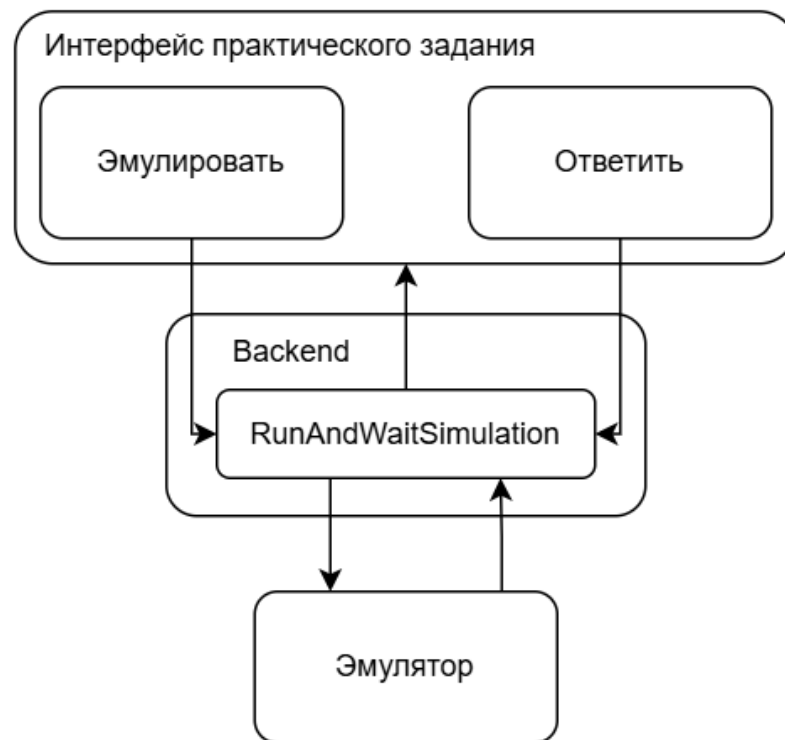


Рис. 9: Схема проверки ответа пользователя в исходной архитектуре

Поскольку проверка основывалась исключительно на пользовательской конфигурации, потребовалось разработать дополнительный слой, позволяющий запускать независимые проверки поверх пользовательской сети. Новый промежуточный модуль обеспечил возможность:

- игнорировать пользовательские настройки пинга и подставлять собственные тестовые сценарии;
- генерировать модификации исходной сети для проверки устойчивости к изменениям;
- сохранять результаты проверки в базу данных без отображения их пользователю.



Рис. 10: Схема проверки ответа пользователя в модифицированной архитектуре

Последнее требование было особенно важно для компании Kornfield, так как в рамках отбора кандидатов необходимо было скрывать промежуточные баллы, чтобы исключить возможность повторного прохождения тестов и недобросовестного поведения посредством перепрохождения задания, пока не наберётся нужное количество баллов.

Для реализации этих изменений была расширена архитектура подсистемы: добавлены механизмы подстановки команд, автоматического формирования дополнительных конфигураций и асинхронной записи результатов. Внесённые доработки подтвердили гибкость уже предложенной архитектурной модели и её способность адаптироваться к новым прикладным требованиям.

4. Особенности реализации

4.1. Автоматическая проверка практических задач

4.1.1. Типы реализованных проверок сетевых конфигураций

В процессе работы над подсистемой проверки практических заданий была существенно расширена часть, определяющая правила интерпретации пользовательских решений и критерии корректности. В предыдущей версии подсистемы проверка основывалась исключительно на прохождении ICMP-пакета от начального хоста к конечному. В рамках этапа разработки были внедрены новые возможности алгоритма проверки, позволяющие точнее формализовать условия практического задания.

В частности, введены следующие типы проверок:

- проверка приватности IP-адресов;
- проверка минимального значения VLAN ID;
- проверка прохождения ring между хостами (в том числе по заданному маршруту и с учетом направления (одностороннее/двустороннее/различными путями));
- проверка IP и маски подсети на интерфейсах;
- проверка принадлежности устройств к одной или разным VLAN;
- отсутствие или наличие default gateway на хосте;
- туннелирование ICMP-пакетов (IPIP, GRE, VxLAN);
- совпадение IP-адресов двух устройств между собой;
- совпадение IP-адреса устройства с заданным значением.

Также была реализована гибкая система начисления баллов, учитывающая выполнение каждого из этих отдельных условий и позволяющая точно настраивать оценку задания. Получение дифференцированного балла можно увидеть на рисунке 11.

4.1.2. Учебный и контрольный режимы

Учебный режим предоставляет пользователям подсказки, какие условия задания они не выполнили, в случае получения неполного балла за задачу.

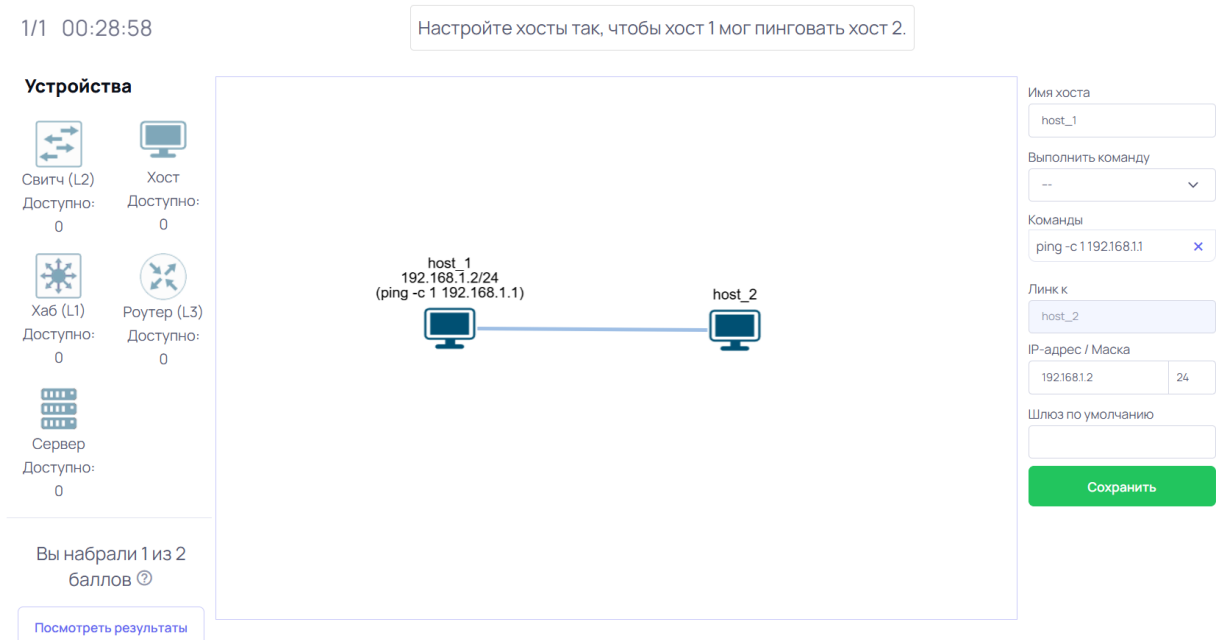


Рис. 11: Получение пользователем неполного балла за задачу.

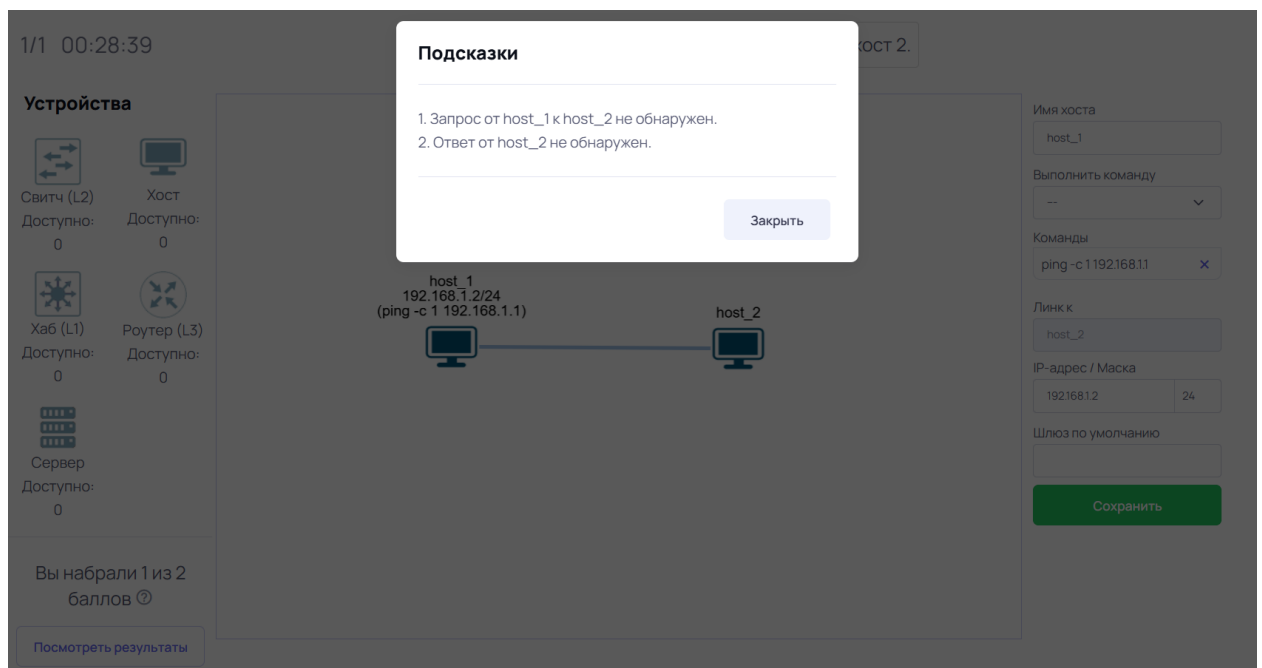


Рис. 12: Подсказки к заданию.

Контрольный режим представлен в двух модификациях: с моментальным и отложенным открытием результата пользователю.

✔ **Этап 1 (начальный уровень)**

Связность H1 и S1 (должен проходить ping) и связность между H2 и H3. Поток трафика от R2 до R3 должен проходить через сеть N23, поток трафика от R3 до R2 должен проходить через сеть N32. При выходе из строя одной из сетей N23 или N32 связь не должна нарушаться.

1 вопрос
До 07.05.2025, 23:59:00

Рис. 13: Раздел с отложенным открытием результатов.

В случае прохождения контрольной работы с моментальным открытием результата опыт пользователя похож на опыт в учебном режиме, кроме того, что ему не предоставляются подсказки. Кроме того, такая контрольная работа всегда ограничена по времени. Этот формат подходит для проведения зачётов по компьютерным сетям.

Miminet

Контрольный тест / Раздел 1

Тестирование Примеры сетей Мои сети

1/3 00:29:03

Настройте сеть так, чтобы хост 1 пинговал хост 2. Их широковещательные пакеты не должны приходить на хост 3 и хост 4.

Устройства

Свитч (L2)

Доступно: 2

Хост

Доступно: 2

Хаб (L1)

Доступно: 2

Роутер (L3)

Доступно: 2

Сервер

Доступно: 2

Вы набрали 4 из 5 баллов

Следующий вопрос

host_1
10.0.0.1/24
(ping -c 1 10.0.0.2)

host_2
10.0.0.2/24

host_3

host_4

l2sw1

l2sw2

Тут будут настройки устройств. Выделите любое на схеме.

Рис. 14: Интерфейс контрольной работы. Результат выводится сразу, подсказки не предоставляются.

В контрольном режиме с отложенным открытием результата пользователь не получает результат сразу же, а просто продолжает контрольную работу.

27

1/1

Этап включает в себя все условия из этапа 1. А также:

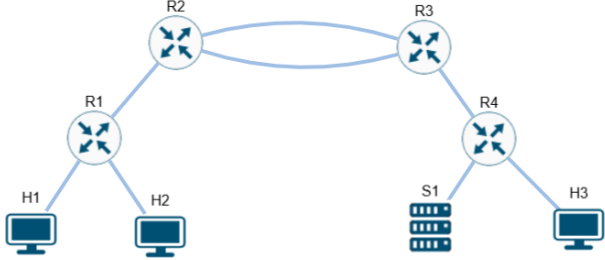
- Настроить маршрутизаторы R1 и R4 таким образом, чтобы сети H1 ← > R1 и S1 ← > R4 не зависели от внутренних сетей провайдера R2 ← > R3.
- Пусть адрес R2 в сети N23 совпадает с адресом H1, а адрес R3 в сети N32 совпадает с адресом S1.

Устройства

 Свитч (L2) Доступно: 0	 Хост Доступно: 0
 Хаб (L1) Доступно: 0	 Роутер (L3) Доступно: 0
 Сервер Доступно: 0	

[На страницу результатов](#)

✓ Ответ сохранён



Тут будут настройки устройств. Выделите любое на схеме.

Рис. 15: Интерфейс контрольной работы с отложенным отображением результата. Результат не выводится, пользователь просто продолжает контрольную работу.

Кроме того, в контрольном режиме с отложенным отображением результата существует возможность не ограничивать время прохождения заданий. Можно увидеть отсутствие таймера на рисунке 15. Кроме того, покидание страницы с заданием до отправки ответа на сервер не завершает сессию контрольной работы, как происходит в других режимах. Это позволяет пользователю продолжить выполнение задания позднее с той же точки, где он закончил. Такой формат хорошо подходит для проведения отбора кандидатов на стажировку.

Таким образом, развитие подсистемы автоматической проверки практических задач позволило не только расширить спектр контролируемых параметров сети (от базовых проверок ICMP до сложных условий вроде туннелирования и валидации VLAN), но и реализовать гибкие форматы оценивания — как в режиме обучения с подсказками, так и в контрольных режимах с различными условиями проверки. Это обеспечивает адаптивность подсистемы под разные учебные сценарии: от

тренировочных заданий до экзаменов и отборочных испытаний.

4.2. Развитие теоретической части подсистемы

Работа над модулем подсистемы, реализующим теоретические задания, была направлена на повышение гибкости формирования тестов и удобства взаимодействия. В первоначальной версии вопросы жёстко привязывались к конкретным тестам, что ограничивало вариативность проверки знаний. Кроме того, существовала возможность недобросовестного подхода в обучении, когда студент мог просто заучить все вопросы к экзамену.

В ходе доработок реализованы следующие ключевые улучшения:

1. внедрена система категорий вопросов, позволяющая автоматически формировать тесты из банка заданий по заданной теме;
2. добавлена возможность прикрепления изображений к вопросам, что особенно важно для визуализации сетевых топологий, схем маршрутизации и других технических аспектов. Для удобства работы с графикой разработан специализированный API, обеспечивающий загрузку и хранение изображений и автоматическую обработку метаданных;
3. разработан API, который позволяет добавлять множество вопросов с помощью одного POST-запроса.

Модернизация модуля с теоретическими заданиями трансформировала его из статичного набора тестов в адаптивную систему, способную динамически генерировать задания под конкретные учебные цели. Это особенно востребовано при подготовке вариативных контрольных работ или проведении массовых тестирований (например, для оценивания студентов на зачёте).

4.3. Улучшение пользовательского интерфейса

Работа над пользовательским интерфейсом системы была направлена на повышение прозрачности процесса обучения и удобства навигации между различными режимами работы. В предыдущей версии интерфейса не обеспечивал четкого разделения между результатами теоретической и практической частей, а также не позволял легко анализировать выполненные задания.

Ключевые улучшения перечислены ниже.

1. Новые экраны результатов.

- Четкое разделение результатов теоретического тестирования и практических заданий в краткой сводке по итогам выполнения теста.

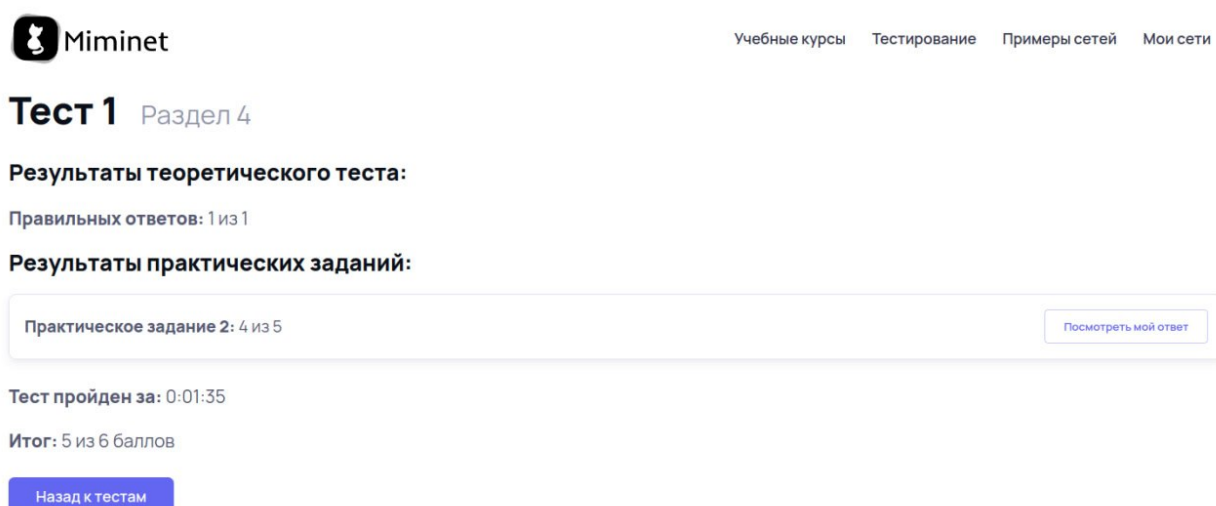


Рис. 16: Интерфейс результатов в виде краткой сводки с возможностью просмотра детальной информации по каждому заданию.

- Возможность просмотра детальной информации по каждому практическому заданию.
- Доработан экран результатов с подробным описанием каждого теоретического и практического вопроса.

Тест 1 Этап 2

Результаты теста

✓ Какой размер MAC-адреса в Ethernet

Баллы: 1 из 1

○ Условия:

- Связность H1 и S1 (должен проходить ping) и связность между H2 и H3.
- Поток трафика от R2 до R3 должен проходить через сеть N23, поток трафика от R3 до R2 должен проходить через сеть N32.
- При выходе из строя одной из сетей N23 или N32 связь не должна нарушаться.

[Посмотреть мой ответ](#)

Баллы: 5 из 9

Тест пройден за: 0:14:21

Итоговый результат: 6 / 10 баллов

[Назад к тестам](#)

Рис. 17: Интерфейс подробных результатов.

- Напоминание о дате открытия результатов, в случае если они ещё недоступны.

Yadro Impulse Этап 2

⌚ Ответы появятся позже

Вы завершили экзамен. Результаты станут доступны 07.05.2025 23:59.

Условия:

- Связность H1 и S1 (должен проходить ping) и связность между H2 и H3.
- Поток трафика от R2 до R3 должен проходить через сеть N23, поток трафика от R3 до R2 должен проходить через сеть N32.
- При выходе из строя одной из сетей N23 или N32 связь не должна нарушаться.

[Посмотреть мой ответ](#)[Назад к тестам](#)

Рис. 18: Интерфейс ещё недоступных результатов.

2. Визуальное выделение учебного и контрольного режимов с помощью специальных значков.

Учебный

✓ Раздел 1	1 вопрос 30 минут
✓ Раздел 2	1 вопрос Тестирование завершено
✓ Раздел 3	1 вопрос До 05.05.2025, 23:00:00
□ Раздел 4	2 вопроса 30 минут

Учебное задание: этот раздел предназначен для обучения. При неверном выполнении задания будут даны разъяснения.

Рис. 19: Отображение учебного и различных контрольных режимов.

Модернизация пользовательского интерфейса упростила процесс навигации для пользователей. Новый экран результатов позволяет быстро оценить свои успехи или продемонстрировать их преподавателю, как в теоретической, так и в практической части. Возможность просмотра ранее настроенных сетей дает ценный инструмент для анализа ошибок и совершенствования навыков работы с сетевыми конфигурациями.

5. Апробация

С апреля 2025 г. 149 уникальных пользователей выполнили практические задания с автоматической проверкой в общей сложности 926 раз. В ходе работы участники оставляли отзывы о найденных ошибках в посвящённом сообществу Miminet telegram-чате [6]: в частности, была выявлена и оперативно исправлена проблема, мешавшая настройке VxLAN в режиме выполнения практического задания.

Кроме того, при отборе на летнюю стажировку Yadro Impulse 2025 подсистемой тестирования воспользовались 87 раз, из них 73 задачи были завершены и отправлены на оценку. Алгоритм проверки формировал подробные комментарии для экспертов, чтобы они могли видеть, за что были сняты или начислены баллы, и после нескольких итераций улучшений подсистемы его результаты работы стали полностью совпадать с оценками проверяющих.

Заключение

В ходе выполнения данной работы были получены следующие результаты.

1. На основе анализа потребностей преподавателей Санкт-Петербургского государственного университета, студентов курса "Основы компьютерных сетей" на платформе Stepik, а также запросов команды Kornfield из компании Yadro, были определены ключевые направления для улучшения системы. В частности, выявлена необходимость в добавлении новых типов заданий и улучшении пользовательского интерфейса.
2. Доработана архитектура подсистемы тестирования, включающая в себя модули для поддержки практических заданий и API для добавления заданий администратором. Определены технологии для реализации: Flask (для API), Bootstrap (для интерфейса) и SQLite и SQLAlchemy (для работы с базой данных). Определена дорабатываемая функциональность.
3. Разработана новая функциональность.
 - Реализованы модули проверки практических заданий для проверки обучающихся, среди которых проверка настройки IP-адресов, VLAN'ов, отправки ICMP-пакетов с различными параметрами и default gateway.
 - Разработаны учебный и контрольный режимы, позволяющие пользователям как обучаться, получая обратную связь в случае неверного выполнения задания, так и выполнять задания без подсказок, чтобы получить оценку или пройти сертификацию.
 - Доработан пользовательский интерфейс (модальные окна и иконки внутри практических заданий, просмотр результатов теста, возможность добавлять картинки в теоретические вопросы и т.д).

- Добавлена возможность генерации учебных тестов из случайных вопросов заданной категории, вместо жесткой привязки конкретных вопросов к конкретному разделу.
4. Проведена апробация при отборе кандидатов на летнюю стажировку Yadro Impulse 2025.

Код работы доступен на GitHub [7].

Список литературы

- [1] СПбГУ. Веб-эмулятор Miminet. — URL: <https://miminet.ru/>.
- [2] Страница стажировки YADRO ИМПУЛЬС 2025. — URL: <https://careers.yadro.com/impulse/>.
- [3] URL: <https://yadro.com/ru/kornfeld>.
- [4] Курс «Основы компьютерных сетей». — URL: <https://stepik.org/course/208904/syllabus>.
- [5] URL: <https://github.com/mimi-net/miminet/commit/51c02e05c1bdd7d62f780f3f1051035c8fe38905>.
- [6] Miminet Community. — URL: <https://t.me/MimiNetCommunity>.
- [7] URL: <https://github.com/sayumikko/miminet>.
- [8] Rogério Leão Santos De Oliveira, Christiane Marie Schweitzer, Ailton Akira Shinoda, Ligia Rodrigues Prete // Using mininet for emulation and prototyping software-defined networks. — 2014 IEEE Colombian conference on communications and computing (COLCOM), 2014. — P. 1–6.
- [9] Authors SQLAlchemy. SQLAlchemy: The Database Toolkit for Python. — URL: <https://www.sqlalchemy.org/> (дата обращения: 2025-02-24). ORM-библиотека для работы с базами данных.
- [10] Docker Inc. Docker Documentation. — URL: <https://docs.docker.com/>.
- [11] Foundation Mozilla. JavaScript | MDN Web Docs. — URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [12] Hipp D. Richard. SQLite Home Page. — URL: <https://www.sqlite.org/>.

- [13] Javid Sheikh Raashid // Role of packet tracer in learning computer networks. — International Journal of Advanced Research in Computer and Communication Engineering, 2014. — Vol. 3. — P. 6508–6511.
- [14] Lal N Dayanand, Ghorbani Behnam, Vaghri Solmaz // A survey on the use of GNS3 for virtualizing computer networks. — 2016.
- [15] Team Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library. — URL: <https://getbootstrap.com/>.
- [16] Team Pallets. Flask Documentation. — URL: <https://flask.palletsprojects.com/en/3.0.x/>.
- [17] Team Pallets. Jinja2 Documentation. — URL: <https://jinja.palletsprojects.com/en/2.10.x/>.
- [18] jQuery Foundation The. jQuery: The Write Less, Do More, JavaScript Library. — URL: <https://jquery.com/>.
- [19] Бакова Елена Зауровна // Проектирование и реализация сетевого тренажера для обучения компьютерным сетям на базе веб-сервиса Miminet. — URL: https://se.math.spbu.ru/thesis/texts/Bakova_Elena_Zaurovna_Bachelor_Thesis_2024_text.pdf.
- [20] Киселева Елизавета Александровна // Обзор программных средств для обучения специалистов в области сетевых технологий. — Постулат., 2020.