Федеральное государственное автономное образовательное учреждение высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

Направление подготовки/ специальность: Разработка и интеграция бизнес-приложений

ОТЧЕТ

по проектной практике

Студент: Бензая Яна Семеновна; Группа: 241-362

Студент: Дзантиев Азамат; Группа: 241-362

Место прохождения практики: Московский Политех, кафедра «Инфокогнитивные технологии»

Отчет принят с оценкой _____ Дата _____

Руководитель практики: Кулибаба Ирина Викторовна

Москва 2025

Оглавление

ОБЩАЯ ИНФОРМАЦИЯ О ПРОЕКТЕ

Современный рынок труда предъявляет всё более высокие и быстро меняющиеся требования к специалистам. Компании ищут не просто дипломированных выпускников, а сотрудников, обладающих конкретным набором профессиональных и универсальных компетенций, способных к обучению, адаптации и решению практических задач. Однако во многих случаях выпускники вузов испытывают затруднения при выходе на рынок труда: они не до конца осознают свои сильные и слабые стороны, не могут грамотно представить свои навыки работодателю, а образовательные программы не всегда обеспечивают развитие актуальных для отрасли компетенций.

Проект «Карты компетенций для выпускника» направлен на решение этих проблем. Его основная цель — создать структурированные и наглядные карты компетенций для студентов по направлениям *инноватика* и *прикладная информатика*, ориентируясь на реальные требования работодателей. Эти карты станут инструментом для осознанного построения образовательной и профессиональной траектории, а также будут полезны вузам для совершенствования программ обучения и работодателям — для подбора подходящих кандидатов.

Для достижения цели проекта «Карты компетенций для выпускника» предусмотрен комплекс задач, направленных на создание полезного и практико-ориентированного инструмента для студентов, вузов и работодателей.

Первым этапом стало проведение детального анализа Федеральных государственных образовательных стандартов (ФГОС) по направлениям инноватика и прикладная информатика, на основе которого был сформирован перечень ключевых профессиональных и универсальных компетенций. Далее разрабатывались содержательные описания каждой компетенции с акцентом на их прикладное значение, применимость в реальных рабочих ситуациях и соответствие актуальным требованиям работодателей.

Важным направлением работы стало определение и подбор эффективных методов развития компетенций. Для каждой из них были подобраны практические

инструменты: курсы, тренинги, проектные задачи, внешние ресурсы, а также рекомендации по саморазвитию. Эти материалы позволяют студентам планировать развитие конкретных навыков, необходимых для профессионального роста.

Для повышения доступности и наглядности была создана инфографика, визуализирующая структуру и взаимосвязи компетенций. Визуальные решения стали частью общей лендинговой страницы проекта, где в открытом доступе представлены карты компетенций, описания, рекомендации и инструменты развития.

Особое внимание уделено связке компетенций с реальными вакансиями на рынке труда: были проанализированы актуальные предложения работодателей и определено, какие навыки наиболее востребованы в различных профессиональных сферах. Это позволило дополнить карты компетенций примерами должностей и требований к соискателям.

Дополнительно в рамках проекта разрабатывается Telegram-бот, который помогает студентам в развитии компетенций. Пользователь выбирает интересующую его компетенцию, а бот выдает подборку полезных ресурсов, курсов, упражнений и других материалов, направленных на развитие именно этого навыка. Такой подход делает процесс обучения и саморазвития более удобным, персонализированным и интерактивным.

Таким образом, задачи проекта охватывают полный цикл — от анализа стандартов и требований рынка труда до создания доступного цифрового инструмента, помогающего студентам осознанно выстраивать свою образовательную и карьерную траекторию.

Актуальность проекта подтверждается статистикой: по данным ТАСС, в ноябре 2024 года уровень безработицы в России составил 2,3%, из которых почти пятая часть — молодёжь до 25 лет. Развитие и осознание собственных компетенций — важный шаг к повышению уровня занятости среди молодых специалистов и формированию устойчивой профессиональной позиции на рынке труда.

Проект «Карты компетенций для выпускника» — это не только инструмент карьерной навигации для студентов, но и шаг к сближению системы образования с реальными потребностями экономики.

1 ОРГАНИЗАЦИЯ

Заказчиком проекта «Карты компетенций для выпускника» является Автономная некоммерческая организация «Россия — страна возможностей» — крупная федеральная платформа, созданная по инициативе Президента Российской Федерации. Её миссия заключается в создании условий для самореализации граждан, поддержки талантливой молодёжи, развития системы социальных лифтов и формирования профессионального потенциала страны.

Организация реализует десятки масштабных проектов в сфере образования, карьеры, предпринимательства, добровольчества и профориентации. Среди них — такие известные инициативы, как «Лидеры России», «Твой ход», «Цифровой прорыв», «Профстажировки» и другие. АНО активно сотрудничает с ведущими университетами, органами государственной власти, компаниями и экспертным сообществом, выстраивая устойчивую экосистему для развития и поддержки молодёжи.

Организационная структура «России — страны возможностей» включает в себя центральный проектный офис, координирующий реализацию программ на федеральном уровне, а также сеть партнёрских организаций, вузов и региональных команд. В рамках работы с университетами особую роль играют Центры оценки и развития компетенций, функционирующие на базе вузов-партнёров по всей России. Эти центры служат точками притяжения для студентов, заинтересованных в профессиональном и личностном развитии, а также пространством взаимодействия между образовательной средой и работодателями.

Центры компетенций предоставляют студентам возможность пройти диагностику своих навыков, получить рекомендации по их развитию, принять участие в тренингах, карьерных мероприятиях и образовательных программах. Они становятся площадками для внедрения современных инструментов оценки и развития компетенций, а также адаптации образовательных программ под реалии и запросы рынка труда.

Проект «Карты компетенций для выпускника» реализуется в рамках этой системной работы по развитию компетенций студентов и является логическим

продолжением миссии АНО «Россия — страна возможностей». Благодаря опоре на опыт организации, её методические ресурсы и партнёрскую сеть, проект получает необходимую экспертизу, технологическую поддержку и возможность масштабирования. Таким образом, заказчик проекта не только инициирует его реализацию, но и создает инфраструктуру, в которой полученные результаты могут быть эффективно внедрены и использоваться в образовательной практике вузов по всей стране.

2 ОПИСАНИЕ ЗАДАНИЙ

2.1 Описание заданий базовой части

Задание на проектную (учебную) практику разработано для студентов первого курса, обучающихся по направлениям подготовки, связанным с информационными технологиями и информационной безопасностью. Трудоёмкость практики составляет 72 академических часа. Задание может выполняться индивидуально или в составе группы до 3 человек. Для управления версиями будет использоваться Git, для написания документации — Markdown, а для создания статического веб-сайта — языки разметки HTML и CSS, но опционально допускается использовать генераторы статических сайтов, такие, как Hugo. В качестве платформы для размещения репозиториев допустимо использовать как GitHub, так и GitVerse, что обеспечивает гибкость в выборе инструментов. Также предусмотрено взаимодействие с организациями-партнёрами, включая стажировки, которые будут приниматься к зачёту при оценке.

1. Настройка Git-репозитория:

- создать групповой репозиторий на GitHub или GitVerse на основе шаблона;
- изучить базовые команды Git;
- регулярно проводить фиксирование изменений с осмысленными сообщениями к коммитам.

2. Написание документов в Markdown:

- все материалы проекта оформить в формате Markdown;
- изучить синтаксис.

3. Создание статического веб-сайта:

- создать сайт с использованием HTML и CSS (или генератора Hugo) по тематике Проектной деятельности;
- включить в сайт следующие страницы: домашняя страница, о проекте, участники, журнал, ресурсы.

4. Взаимодействие с организацией партнером:

- участвовать в профильных мероприятиях;
- подготовить и оформить отчёт о взаимодействии с партнёром в Markdown.

5. Практическая реализация технологии:

- выбрать любую технологию из списка;
- согласовать внутри команды тему, выбрать стек технологий;
- провести исследование, изучение реализации;
- создать подробное описание в формате Markdown;
- создать техническое руководство по созданию проекта;
- модифицировать проект;
- создать видеопрезентацию проекта;
- задокументировать проект в формате Markdown и представить его на сайте.

6. Итоговый отчёт:

- составить отчет по проектной практике на основе шаблона;
- описать в хронологическом порядке этапы работы;
- представить индивидуальные планы работы;
- загрузить две версии отчета в формате docx и pdf.

2.2 Описание заданий вариативной части

По решению ответственного за проектную (учебную) практику студентам назначается одно из следующих вариативных заданий. Студенты могут направить ответственному свои пожелания по распределению.

Наш выбор:

2. Практическая реализация технологии:

1. Выберите любую технологию (тематику) из списка, представленного в репозитории codecrafters-io/build-your-own-x. По согласованию с ответственными за практику можно использовать другой источник проектов.
2. Согласуйте внутри команды выбранную тему. Выберите стек технологий (подсказки также есть в репозитории).
3. Проведите исследование: изучите, как создать выбранную технологию с нуля, воспроизведите практическую часть.
4. Создайте подробное описание в формате Markdown, включающее:
    - Последовательность действий по исследованию предметной области и созданию технологии.
    - Напишите техническое руководство по созданию этой технологии, ориентированное на начинающих.
    - Включите в руководство:
        - Пошаговые инструкции.
        - Примеры кода.
    - Иллюстрации (картинки, диаграммы, схемы) в количестве от 3 до 10 штук, вставленные в текст для наглядности.
    - Поместите результаты исследования и руководства в общий Git-репозиторий.
5. Создайте техническое руководство или туториал по созданию проекта на выбранную тему. Для визуализации архитектуры, процессов и прочего используйте разные типы диаграмм UML, схемы, графики, таблицы.

6. Сделайте модификацию проекта согласно полученным знаниям и навыкам в течение года (творческий пункт, самостоятельно выбираете в какой части модифицировать). Описать в технической документации модификации.
7. Сделайте видео презентацию выполненной работы (цель, задачи, как решали, демонстрация работоспособного результата).
8. Задокументируйте проект в репозитории в формате Markdown и представьте его на сайте в формате HTML.
9. Подготовить финальный отчет (в хронологической последовательности опишите этапы работы, отдельно должны быть представлены индивидуальные планы каждого участника).

2.3 Описание достигнутых результатов по проектной практике

В рамках базовой части проекта по дисциплине «Проектная деятельность» наша команда последовательно выполнила все предусмотренные этапы. Начали мы с настройки репозитория Git. Был создан групповой репозиторий на GitHub, в который каждый участник получил доступ. Мы освоили базовые команды Git: клонировали репозиторий на локальные машины, создавали ветки, регулярно фиксировали изменения с понятными и осмысленными комментариями к коммитам, а затем отправляли изменения обратно в репозиторий. Это позволило нам эффективно работать над проектом в команде и отслеживать вклад каждого участника.

Затем мы перешли к оформлению проектной документации в формате Markdown. Мы изучили синтаксис Markdown и использовали его для создания описания проекта, журнала прогресса, отчётов, а также технической документации. Markdown-документы размещены в репозитории и сопровождают каждый этап реализации проекта, включая техническое описание и инструкции.

После этого мы приступили к созданию статического веб-сайта. Мы выбрали простой стек HTML и CSS, так как он позволил сфокусироваться на содержании и дизайне, не перегружая проект сложными инструментами. Для оформления использовали уникальный дизайн, не совпадающий с другими работами. Сайт посвящён основной части проекта — разработке и визуализации карт компетенций выпускников по направлениям «Инноватика» и «Прикладная информатика». На сайте представлены:

Домашняя страница с аннотацией проекта — кратко изложены цель, задачи и актуальность.

Страница «О проекте» — описано содержание проекта, этапы, технологии и ожидаемые результаты.

Раздел «Участники» — указаны члены команды и их вклад в реализацию проекта.

Раздел «Журнал» — добавлены три поста о ходе работы, промежуточных результатах и планах.

Страница «Ресурсы» — представлены ссылки на полезные материалы, в том числе на сайты работодателей, использованные стандарты и статьи по теме компетенций.

Визуальная часть сайта была дополнена графикой, оражающей структуру сайта и содержание проекта. Она помогает лучше воспринимать информацию и подчеркивают практическую направленность проекта.

Кроме разработки, мы активно участвовали в профильных мастер-классах и мероприятиях, связанных с тематикой проекта, ездили на экскурсию в офис заказчика. Это позволило лучше понять запросы индустрии, получить обратную связь от специалистов и заказчика и адаптировать проект под реальные требования работодателей. Опыт взаимодействия с организацией-партнёром и опыт посещения профильных мастер-классов мы отразили в отчёте, включённом в наш репозиторий.

Таким образом, все этапы базовой части задания были выполнены: мы настроили Git, подготовили документацию в Markdown, разработали полноценный сайт и установили взаимодействие с профессиональной средой. Каждый этап был задокументирован и отражён в структуре проекта, что обеспечило его целостность, прозрачность и соответствие требованиям.

Вариативной частью проекта стало создание Telegram-бота на JavaScript с использованием Node.js, Express, Axios и body-parser. Нашей задачей было не просто написать бот, который отвечает на команды, а выстроить архитектуру, позволяющую масштабировать проект, добавлять новые функции и сохранять состояние пользователя. Бот умеет обрабатывать команды /start, /menu, /quiz, предлагает инлайн-кнопки и умеет вести простую викторину. Дополнительно реализованы функции развлечения (бот выдаёт анекдоты), а также справочная система (/help). Таким образом, бот выполняет как развлекательную, так и образовательную функции.

Разработка бота позволила нам глубоко погрузиться в технологии создания телеграм-сервисов, построить систему с запоминанием состояний и расширяемой логикой. Результат был оформлен в виде технического руководства и пошагового туториала: для новичков мы подготовили понятное руководство, а для разработчиков — подробное описание архитектуры. Благодаря этому наш проект стал платформой, на базе которой можно строить другие телеграм-приложения — от образовательных до бизнес-инструментов.

Вся работа велась через GitHub, где мы освоили основные команды Git: коммиты, ветвление, пуш, создание pull-запросов. Вся документация была оформлена в Markdown: описания, руководства, отчёты, журнал прогресса. Мы добились того, чтобы проект был не просто технически реализован, но и грамотно задокументирован и представлен в понятной, доступной форме.

Таким образом, проект позволил нам применить полученные знания на практике. Мы получили ценный опыт командной работы, технической реализации, взаимодействия с профессиональной средой и представления проекта в форме, соответствующей современным требованиям к цифровым продуктам.

3 ИНДИВИДУАЛЬНЫЕ ПЛАНЫ УЧАСТНИКОВ

3.1 Дзантиев Азамат

Задача	Время, ч
Клонирование репозитория, заполнение его по заданному шаблону	1.5
Изучение работы с GitHab	4,5
Изучение настроек Git	7
Изучение синтаксиса Markdown	4
Изучение HTML	5
Взаимодействие с организацией-партнером «ООО Ингосстрах»	2
Взаимодействие с организацией-партнером «ООО Россия – Страна возможностей»	5
Изучение и настройка «Hugo»	12
Наполнение сайта	9
Исследование технологии «Node.js: How to make a responsive telegram bot»	6
Создание руководства по созданию выбранной технологии, ориентированное на начинающих	6
Создание технического руководства и по созданию проекта на выбранную тему	6
Модификация проекта	4
Представление информации о проекте вариативной части на сайте (HTML)	4
Видео презентация выполненной работы	4

Итого: 76 часов.

3.2 Бензая Яна

Задача	Время, ч
Изучение работы с GitHab	4,5
Изучение настроек Git	7
Изучение синтаксиса Markdown	4
Взаимодействие с организацией-партнером «Yandex»	5
Написание отчета о взаимодействии с партнером «Yandex»	4
Взаимодействие с организацией-партнером «ООО Ингосстрах»	2
Написание отчета о взаимодействии с организацией-партнером «ООО Ингосстрах»	4
Взаимодействие с организацией-партнером «ООО ПСБ»	2
Написание отчета о взаимодействии с организацией-партнером «ООО ПСБ»	4
Взаимодействие с организацией-партнером «ООО Россия – Страна возможностей»	5
Написание отчета о взаимодействии с организацией-партнером «ООО Россия – Страна возможностей»	4
Написание материалов проекта	5
Исследование технологии «Node.js: How to make a responsive telegram bot»	6
Создание подробного описания в формате Markdown по исследованию предметной области	4
Создание подробного описания модификации созданного проекта	6

Документирование проекта в репозитории в формате Markdown	8
Создание финального отчёта	10

Итого: 84,5 часов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта мы получили ценный практический опыт, который помог нам не только закрепить теоретические знания, полученные в ходе обучения, но и применить их на практике. Каждый этап проекта — от настройки Git и написания документации в Markdown до создания сайта с использованием Hugo и разработки Telegram-бота — стал для нас возможностью развить технические и организационные навыки, научиться работать в команде и решать реальные задачи.

Мы научились использовать современные инструменты разработки, оформлять проекты по всем требованиям, структурировать информацию и визуализировать результаты. Работа с Hugo, несмотря на возникавшие трудности, позволила нам выйти за рамки базового HTML и познакомиться с принципами генерации статических сайтов. Разработка Telegram-бота также дала нам понимание, как строится интерактивная система с запоминанием состояния и пользовательскими сценариями.

Кроме того, участие в профильных мастер-классах усилило наш интерес к проекту и позволило взглянуть на него с точки зрения реальных потребностей рынка. Благодаря этому мы смогли сделать наш проект более целостным, актуальным и приближенным к реальной практике.

В итоге мы считаем, что успешно справились с задачами, поставленными в рамках проектной деятельности. Мы не только реализовали задуманное, но и получили новые знания, навыки и уверенность в своих силах. Эта практика стала важным шагом в профессиональном развитии каждого из нас. Мы гордимся проделанной работой и уверены, что она имеет практическую ценность.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Hugo: The world’s fastest framework for building websites — The world's fastest framework for building websites (официальная документация генератора статических сайтов Hugo)
2. Hugo Quick Start Guide – Quick start (руководство по быстрой настройке и запуску сайта на Hugo).
3. GitHub Docs – GitHub Docs (для работы с Git, управления репозиториями, совместной разработки и ведения документации).
4. Markdown Guide – Markdown Guide (для написания описаний, отчётов и документации в формате Markdown).
5. Telegram Bot API – Telegram Bot API: полное руководство по созданию ботов / Skillbox Media (руководство по созданию бота).
6. Полное руководство по HTML – HTML Complete Guide – A to Z HTML Concepts | GeeksforGeeks

ПРИЛОЖЕНИЕ

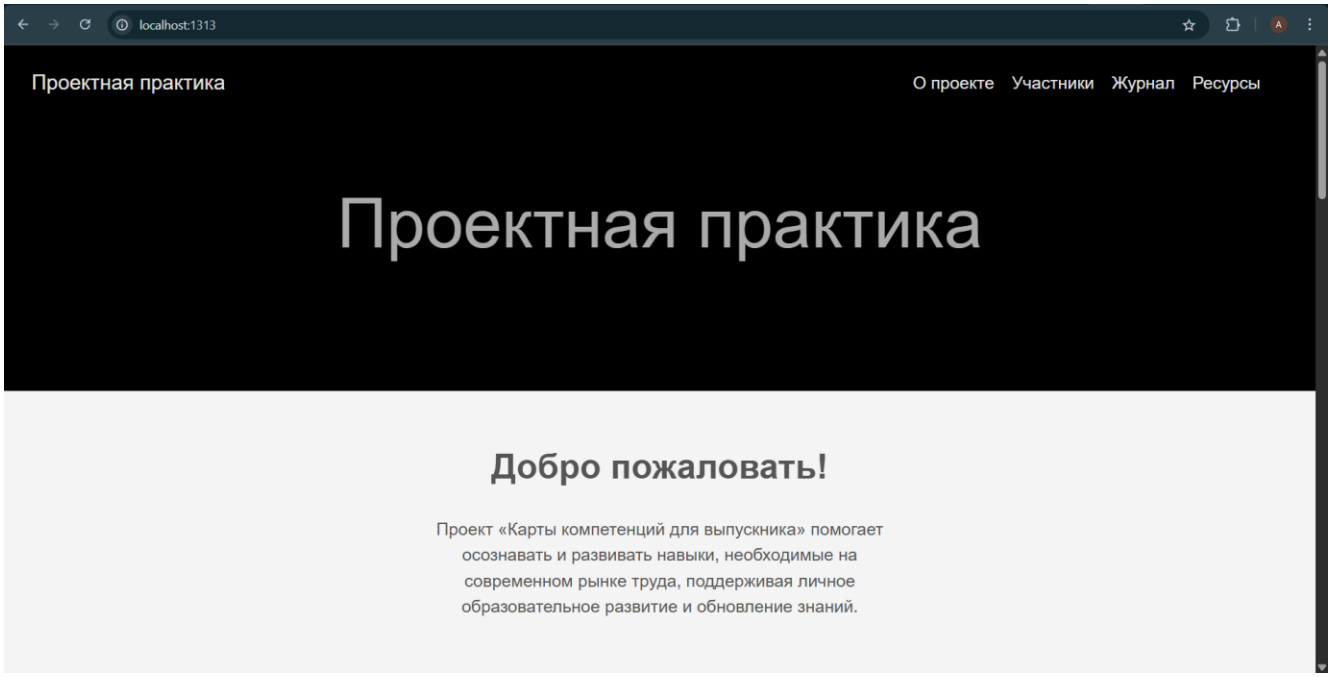


Рисунок 1 - Домашняя страница сайта

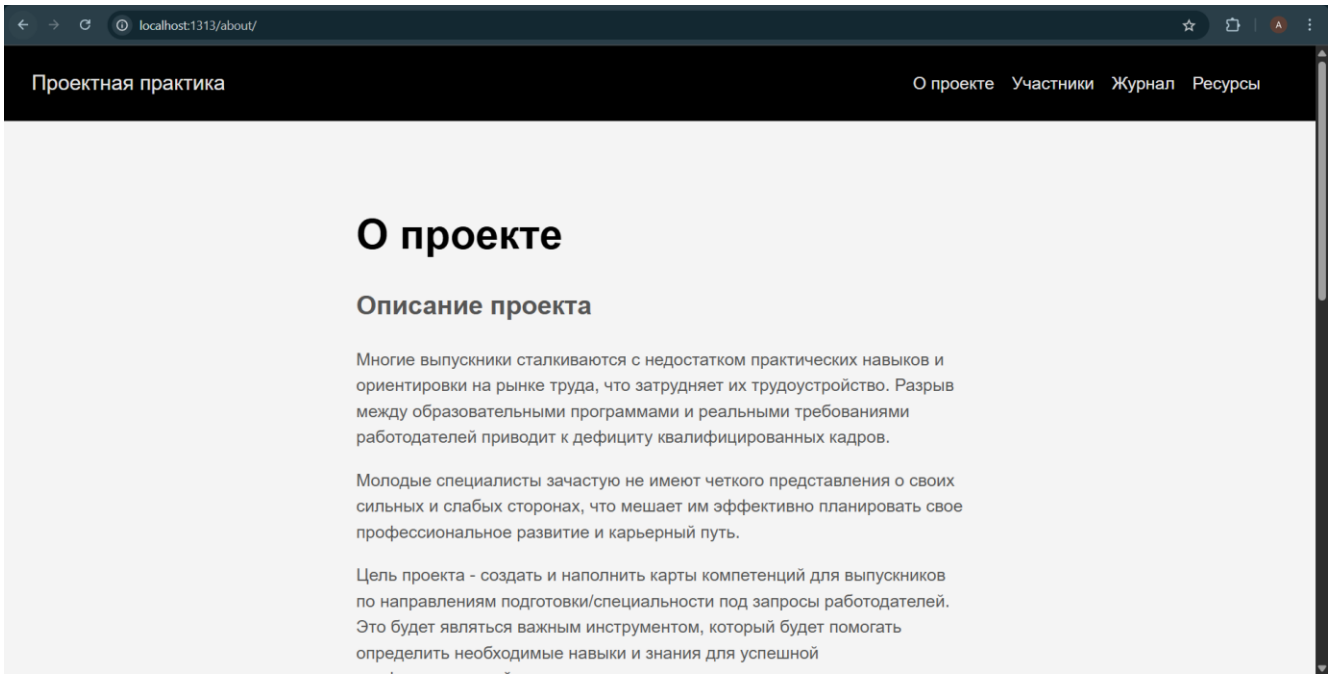


Рисунок 2 - Страница «О проекте»

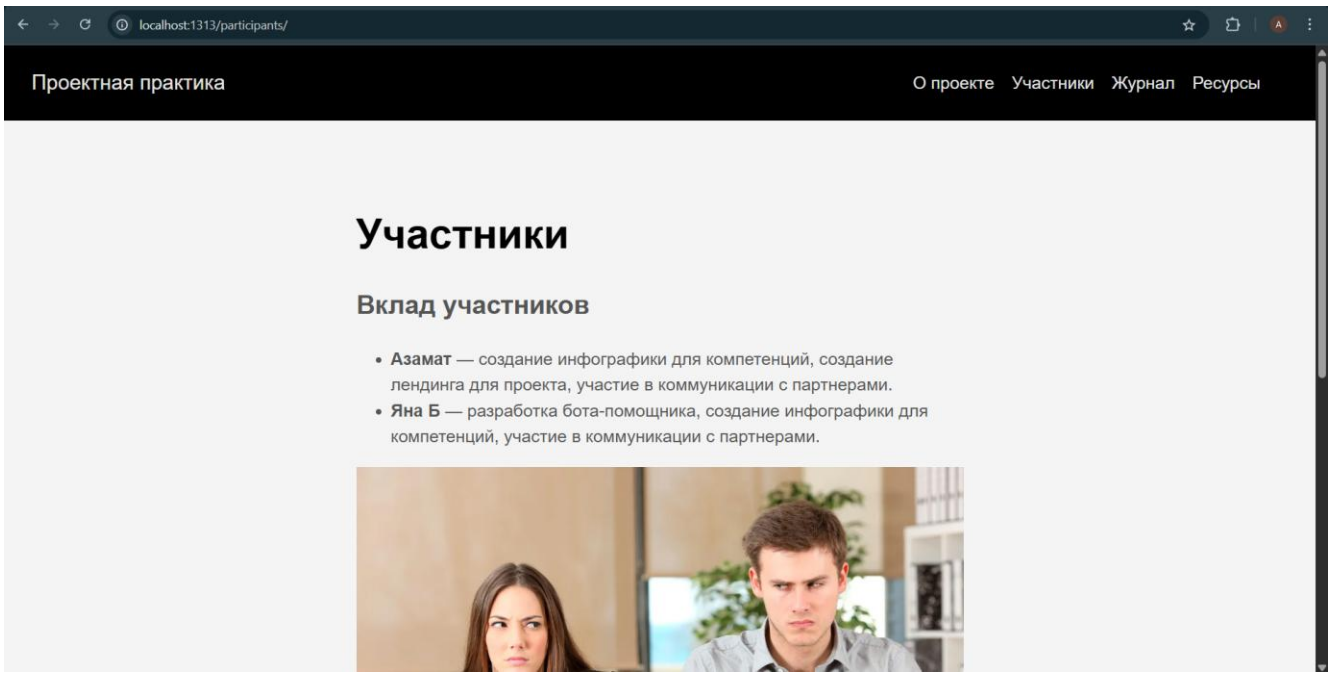


Рисунок 3 - Раздел «Участники»

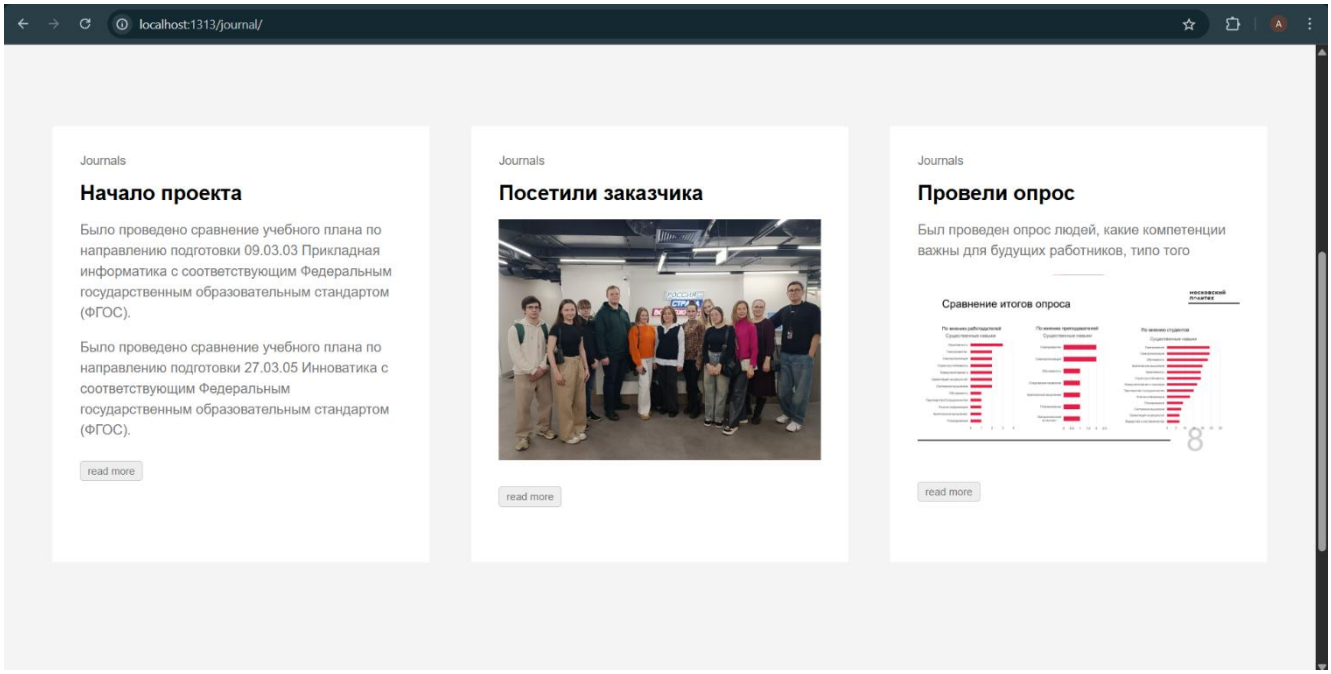


Рисунок 4 - Раздел «Журнал»

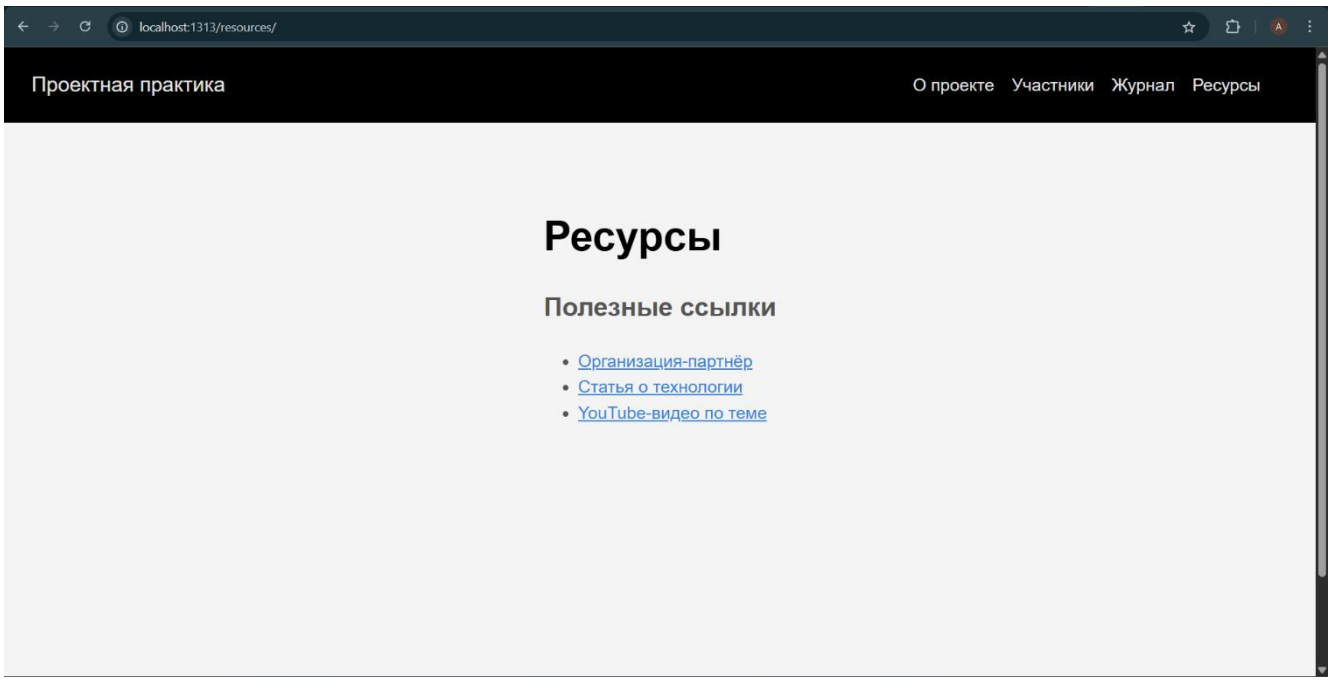


Рисунок 5 - Страница «Ресурсы»

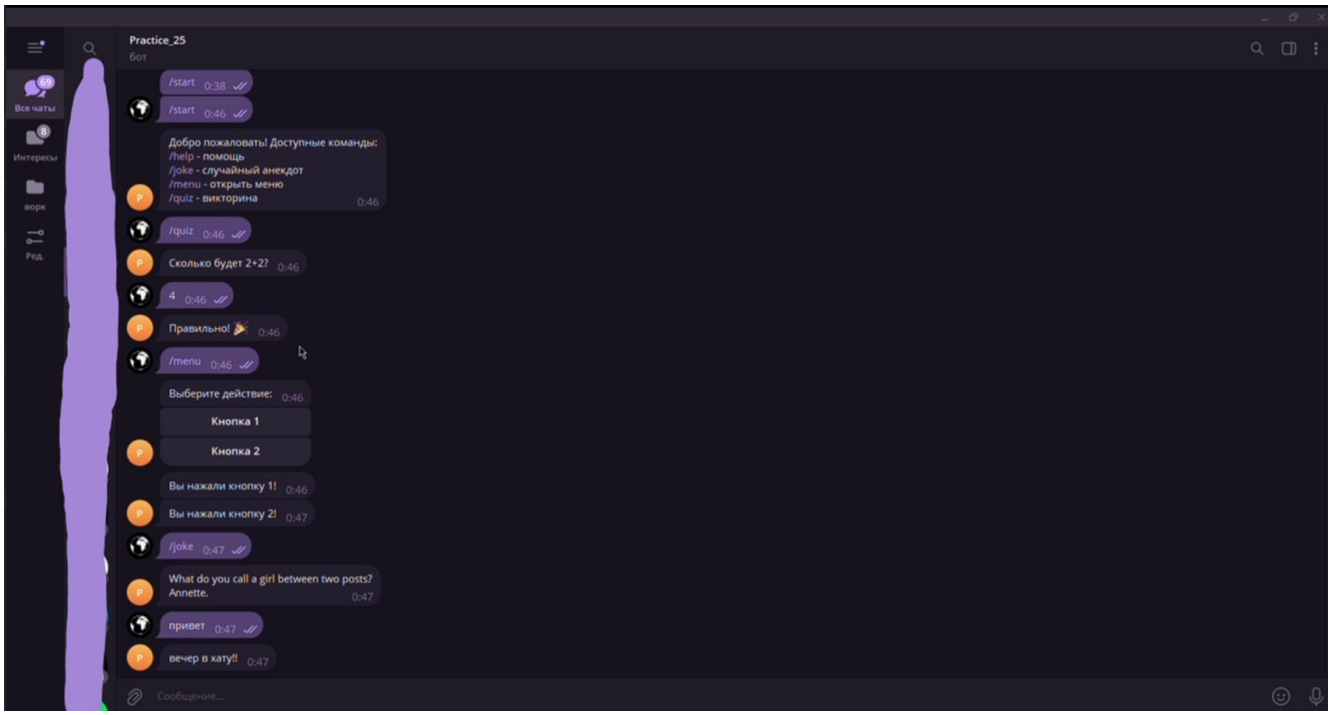


Рисунок 6 - Telegram бот, разарботанный для вариативной части

ОТЧЁТ О ПРОЕКТЕ: Разработка Telegram-бота на Node.js

Введение

Современные мессенджеры давно перестали быть просто средством общения. Они стали полноценными платформами, на базе которых создаются умные ассистенты, игровые механики, системы опросов, интеграции с бизнесом и даже целые образовательные сервисы. Одним из самых популярных инструментов для этих целей является Telegram, благодаря открытому API и поддержке ботов.

В рамках данного проекта мы поставили перед собой задачу — создать Telegram-бота на языке JavaScript (Node.js) с использованием таких инструментов, как Express, Axios и body-parser. При этом цель была не просто сделать «бота, который отвечает», а реализовать интерактивную, расширяемую архитектуру, способную запоминать состояние пользователя, работать с командами и кнопками.

Проект развивался поэтапно: сначала — базовая реализация, затем — постепенное добавление функций, инлайн-кнопок, логики обработки состояний и взаимодействия с пользователем.

Технологический стек

Технология	Назначение
Node.js	Среда выполнения JavaScript
Express	Веб-фреймворк для создания REST-сервера
Axios	Библиотека для HTTP-запросов
body-parser	Парсинг тела POST-запросов
Telegram Bot API	Взаимодействие с Telegram-ботом

Документация и сопровождение проекта

Разработка Telegram-бота — это не только код, но и его понятное описание, чтобы другие участники команды (или даже будущие мы сами) могли быстро вникнуть, поддерживать и дорабатывать систему. Поэтому параллельно с работой над функционалом, мы уделили особое внимание созданию документации.

Руководство по созданию бота для начинающих

- Зарегистрировать бота через BotFather
- Получить и использовать токен
- Настроить Node.js и Express
- Написать базовый обработчик сообщений
- Отправлять простые ответы пользователям
- Обрабатывать команды и кнопки

Это руководство было создано с расчётом на тех, кто никогда не писал ботов и даже, возможно, только начинает свой путь в программировании. Мы включили в него примеры кода, визуальные пояснения, разбор ошибок и советы по отладке. Все шаги были проверены на практике, чтобы читатель не остался наедине с "падает, не знаю почему".

Цель: сделать процесс создания первого бота лёгким, понятным и вдохновляющим.

Руководство по проекту и модификациям

- Архитектура и структура проекта
- Зачем и как мы разделили логику на блоки (обработка сообщений, кнопок, состояний)
- Какие команды реализованы и почему
- Что делает каждая функция
- Идеи для расширения (бот для викторин, справочник, чат-игра и т.д.)

Мы стремились сделать так, чтобы любой разработчик, открывший проект, мог разобраться в нём за 15 минут и сразу начать работу. Это особенно важно в командной разработке, при передаче проекта или при доработке через месяцы.

Подробная модификация: не просто улучшения, а переход на новый уровень

Мы также пошли дальше и добавили в проект расширенную модификацию, о которой говорилось выше. В основном руководстве (для новичков) мы сознательно не вдавались в детали:

- не описывали работу `callback_query`
- не касались хранения состояния пользователя
- не объясняли, как делать пошаговые сценарии

Это было сделано осознанно — чтобы не перегружать начинающего разработчика. Однако в рамках данного отчёта и модификации проекта мы решили:

Разобрать всё детально, шаг за шагом — как создаётся логика бота, как работает инлайн-клавиатура, что такое `userStates` , как обрабатываются различные команды и почему это важно для UX.

Мы не просто улучшили проект — мы перевели его на новый уровень: теперь бот умеет вести диалог, обрабатывать нажатия кнопок, задавать вопросы, запоминать, что делает пользователь, и действовать исходя из контекста.

Архитектура проекта

Структура

Проект построен на простом и понятном сервере Express, который обрабатывает POST-запросы от Telegram, присылаемые при каждом новом сообщении или взаимодействии пользователя с ботом. Ниже показано, как устроена базовая архитектура:

```
Telegram (user) -> POST /new-message -> Обработка сообщения -> Ответ через Telegram Bot API
```

Рисунок 7 - Отчёт в формате html

Реализация

Шаг 1: Базовый сервер

Проект начинается с создания Express-сервера и настройки базового маршрута для приёма сообщений от Telegram.

```
const express = require("express")
const app = express()
const bodyParser = require("body-parser")
const axios = require("axios")

const TOKEN = "ВАШ_ТОКЕН_ТУТ"
const TELEGRAM_API = `https://api.telegram.org/bot${TOKEN}`

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
```

Шаг 2: Обработка входящих сообщений

Мы проверяем, содержит ли входящее сообщение необходимые данные. Если да — анализируем текст и отправляем соответствующий ответ. На первом этапе бот просто реагирует на слово "привет".

```
app.post("/new-message", (req, res) => {
 const { message } = req.body

 if (!message || !message.text || !message.chat) {
   return res.end()
 }

 const text = message.text.toLowerCase()

 if (text.includes("привет")) {
   axios.post(`${TELEGRAM_API}/sendMessage`, {
     chat_id: message.chat.id,
     text: "вечер в хату!!",
   })
   .then(() => res.end("ok"))
   .catch((err) => {
     console.error("Ошибка отправки:", err)
     res.end("Ошибка: " + err)
   })
 } else {
   res.end("ok")
 }
})
```

Бот запускается на порту 3000:

```
app.listen(3000, () => {
  console.log("Сервер запущен на порту 3000")
})
```

Расширенные функции

После успешного запуска базовой версии мы начали поэтапное расширение:

Команда `/start` и справка

```
if (message.text.toLowerCase() === "/start") {
  axios.post(`${TELEGRAM_API}/sendMessage`, {
    chat_id: message.chat.id,
    text: "Добро пожаловать! Доступные команды:\n/help - помощь\n/joke - случайный анекдот",
    parse_mode: "Markdown"
  })
  return res.end()
}
```

- Цель: дать пользователю понимание, что бот умеет делать.
- Особенность: используется Markdown-разметка для форматирования текста.

Обработка состояния пользователя

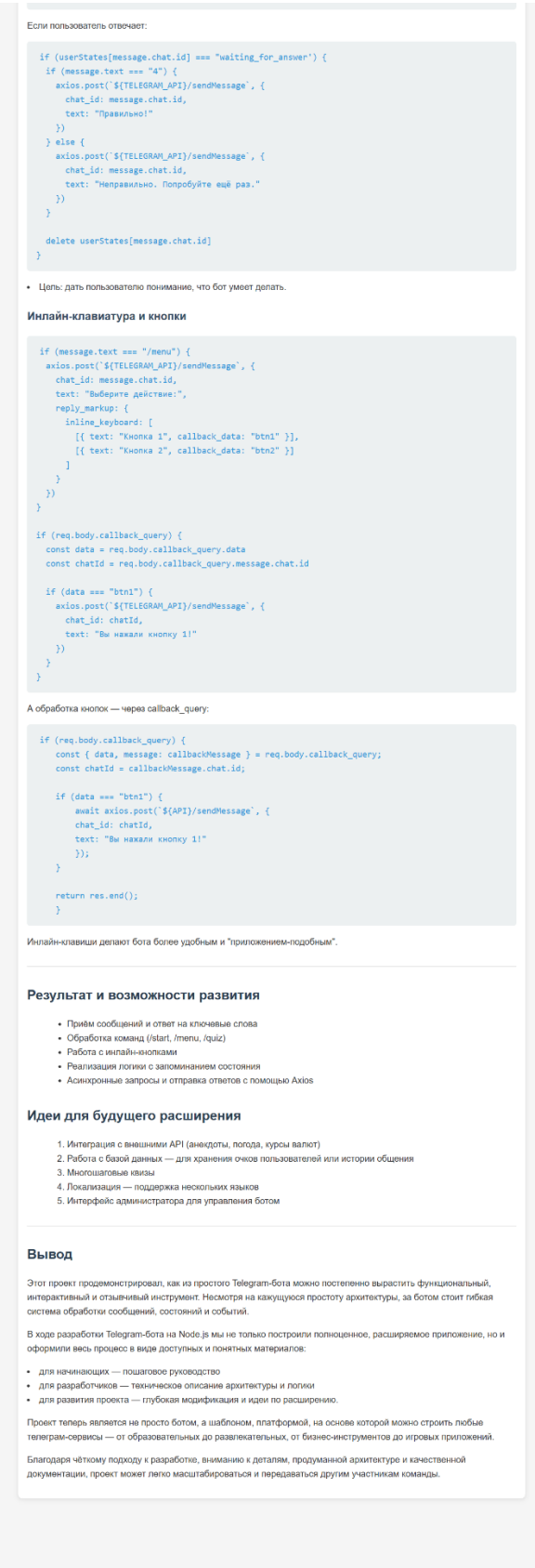
Важным шагом стало внедрение механизма запоминания, что делает пользователь:

```
const userStates = {}

if (message.text === "/quiz") {
  userStates[message.chat.id] = "waiting_for_answer"
  axios.post(`${TELEGRAM_API}/sendMessage`, {
    chat_id: message.chat.id,
    text: "Сколько будет 2+2?"
  })
}
```

Если пользователь отвечает:

Рисунок 8 - Отчёт в формате html



Если пользователь отвечает:

```
if (userStates[message.chat.id] === 'waiting_for_answer') {
 if (message.text === "4") {
   axios.post(`${TELEGRAM_API}/sendMessage`, {
     chat_id: message.chat.id,
     text: "Правильно!"
   })
 } else {
   axios.post(`${TELEGRAM_API}/sendMessage`, {
     chat_id: message.chat.id,
     text: "Неправильно. Попробуйте ещё раз."
   })
 }

 delete userStates[message.chat.id]
}
```

- Цель: дать пользователю понимание, что бот умеет делать.

Инлайн-клавиатура и кнопки

```
if (message.text === "/menu") {
 axios.post(`${TELEGRAM_API}/sendMessage`, {
   chat_id: message.chat.id,
   text: "Выберите действие:",
   reply_markup: {
     inline_keyboard: [
       [{ text: "Кнопка 1", callback_data: "btn1" }],
       [{ text: "Кнопка 2", callback_data: "btn2" }]
     ]
   }
 })
}

if (req.body.callback_query) {
  const data = req.body.callback_query.data
  const chatId = req.body.callback_query.message.chat.id

  if (data === "btn1") {
    axios.post(`${TELEGRAM_API}/sendMessage`, {
      chat_id: chatId,
      text: "Вы нажали кнопку 1!"
    })
  }
}
```

А обработка кнопок — через callback_query:

```
if (req.body.callback_query) {
   const { data, message: callbackMessage } = req.body.callback_query;
   const chatId = callbackMessage.chat.id;

   if (data === "btn1") {
       await axios.post(`${API}/sendMessage`, {
       chat_id: chatId,
       text: "Вы нажали кнопку 1!"
       });
   }

   return res.end();
   }
```

Инлайн-клавиши делают бота более удобным и "приложением-подобным".

Результат и возможности развития

- Приём сообщений и ответ на ключевые слова
- Обработка команд (/start, /menu, /quiz)
- Работа с инлайн-кнопками
- Реализация логики с запоминанием состояния
- Асинхронные запросы и отправка ответов с помощью Axios

Идеи для будущего расширения

1. Интеграция с внешними API (анекдоты, погода, курсы валют)
2. Работа с базой данных — для хранения очков пользователей или истории общения
3. Многошаговые квизы
4. Локализация — поддержка нескольких языков
5. Интерфейс администратора для управления ботом

Вывод

Этот проект продемонстрировал, как из простого Telegram-бота можно постепенно вырастить функциональный, интерактивный и отзывчивый инструмент. Несмотря на кажущуюся простоту архитектуры, за ботом стоит гибкая система обработки сообщений, состояний и событий.

В ходе разработки Telegram-бота на Node.js мы не только построили полноценное, расширяемое приложение, но и оформили весь процесс в виде доступных и понятных материалов:

- для начинающих — пошаговое руководство
- для разработчиков — техническое описание архитектуры и логики
- для развития проекта — глубокая модификация и идеи по расширению.

Проект теперь является не просто ботом, а шаблоном, платформой, на основе которой можно строить любые телеграм-сервисы — от образовательных до развлекательных, от бизнес-инструментов до игровых приложений.

Благодаря чёткому подходу к разработке, вниманию к деталям, продуманной архитектуре и качественной документации, проект может легко масштабироваться и передаваться другим участникам команды.

Рисунок 9 - Отчёт в формате html

Ссылка на GitHub репозиторий :

[WhaleeAi/pr_25]