**Исследовательский план и дорожная карта обучения для проекта «GrindStone»**

**Цель:** Освоить полный стек технологий, необходимых для самостоятельной разработки и развертывания веб-приложения «GrindStone».  
**Срок:** ~6-9 месяцев (при интенсивном регулярном обучении).  
**Исполнитель:** Мамаев Артём Евгеньевич.

**Этап 0: Фундаментальная подготовка (~ 1 месяц)**

**Цель:** Понимать базовые принципы веб-разработки.

* **HTML5:** Семантическая верстка, основные теги (<section>, <article>, <div>, формы).
* **CSS3:** Flexbox, Grid, позиционирование, адаптивная верстка (медиа-запросы). *Не просто знать свойства, а уметь верстать макеты.*
* **JavaScript (ES6+):** **Ключевая тема!**
  + Синтаксис: переменные (let, const), типы данных, операторы.
  + Функции (обычные, стрелочные).
  + Массивы и методы массивов (map, filter, reduce).
  + Объекты.
  + Асинхронность: Promise, async/await (критически важно для работы с API).
  + Fetch API для работы с HTTP-запросами.
* **Git и GitHub:** Базовые команды (git init, add, commit, push, pull). Создание репозитория для проекта, ведение истории изменений.

**Результат этапа:** Создан простой статический сайт-портфолио с адаптивной версткой, размещенный на GitHub Pages.

**Этап 1: Погружение во Фронтенд (~ 2-3 месяца)**

**Цель:** Освоить фреймворк Next.js и сопутствующие инструменты.

* **React (базовый):**
  + Концепция компонентов (функциональные компоненты).
  + JSX-синтаксис.
  + Хуки: useState (состояние), useEffect (жизненный цикл), useContext.
  + Пропсы (props) и передача данных между компонентами.
* **Next.js:**
  + **Почему он первый?** Он дает структуру и решает сложные задачи (роутинг, рендеринг) из коробки.
  + File-based Routing (роутинг на основе файловой структуры).
  + Pre-rendering: Static Generation (SSG) и Server-Side Rendering (SSR). *Это ключ к быстрой загрузке, как в Notion.*
  + Создание страниц и layouts.
* **Tailwind CSS:**
  + Изучение утилитарного подхода. Быстрое прототипирование и создание UI без написания кастомного CSS.
* **TypeScript (базовый):**
  + Базовая типизация: типы для переменных, функций, пропсов компонентов.
  + Интерфейсы для описания сложных объектов (например, задача, проект).

**Результат этапа:** Созданное на Next.js простое SPA-приложение (например, блог) с использованием хуков, типизации TypeScript и стилизованное с помощью Tailwind CSS. Компоненты разбиты правильно.

**Этап 2: Освоение Бэкенда (~ 2-3 месяца)**

**Цель:** Научиться создавать robust (надежный) и структурированный API.

* **Node.js:** Основы работы с сервером на JavaScript.
* **Nest.js:**
  + **Почему Nest, а не Express?** Архитектура из коробки (модули, сервисы, контроллеры) научит вас правильной структуре проекта с самого начала.
  + Создание модулей (Modules), контроллеров (Controllers), сервисов (Services).
  + Внедрение зависимостей (Dependency Injection) — ключевая концепция фреймворка.
  + Работа с входящими данными (DTO - Data Transfer Objects), валидация.
* **REST API:**
  + Проектирование эндпоинтов (URLs) для сущностей projects, tasks, auth.
  + HTTP-методы (GET, POST, PUT, DELETE) и коды ответов.
* **Базы данных и ORM:**
  + **SQL и PostgreSQL:** Основы реляционных БД: таблицы, связи (один-ко-многим, многие-ко-многим), запросы (SELECT, INSERT, UPDATE, JOIN).
  + **Prisma:** Самый современный ORM.
    - Схема данных (schema.prisma): описание моделей User, Project, Task.
    - Генерация Prisma Client.
    - Выполнение CRUD-операций (Create, Read, Update, Delete) из кода Nest.js.
* **Аутентификация и авторизация:**
  + **JWT (JSON Web Tokens):** Принцип работы.
  + Реализация регистрации, входа и защиты маршрутов.
  + **NextAuth.js:** Интеграция с Next.js для упрощения процесса на фронтенде.

**Результат этапа:** Созданное REST API на Nest.js для одной сущности (например, «Задачи») с полным CRUD, подключенное к PostgreSQL через Prisma, с простой JWT-аутентификацией.

**Этап 3: Интеграция и Продвинутые темы (~ 1-2 месяца)**

**Цель:** Связать фронтенд и бэкенд, реализовать ключевой функционал.

* **Взаимодействие Frontend и Backend:**
  + Написание запросов с фронтенда (Next.js) к бэкенду (Nest.js) с использованием fetch или библиотеки axios.
  + Обработка состояний загрузки и ошибок.
* **Реализация основных функций GrindStone:**
  + Drag-and-Drop для Kanban-досок (библиотека react-beautiful-dnd или @dnd-kit).
  + Загрузка файлов (библиотека multer для Nest.js, хранение на сервере или в облаке, например, AWS S3).
  + WYSIWYG-редактор для вики-страниц (например, Tiptap или Editor.js).
  + Глобальный поиск (реализация на бэкенде с помощью ILIKE запросов в PostgreSQL или полнотекстового поиска).
* **Деployment (Развертывание):**
  + **Docker:** Создание Dockerfile и docker-compose.yml для контейнеризации приложения и БД.
  + **Хостинг:** Развертывание на VPS (например, Selectel) или платформе (Railway, Heroku). Покупка домена, настройка Nginx.

**Результат этапа:** Полноценное, хотя и минимальное, рабочее веб-приложение, развернутое в интернете, где можно зарегистрироваться, создавать проекты и задачи.

**Этап 4: Качество, Безопасность, Финальная полировка**

**Цель:** Научить код профессиональным стандартам.

* **Тестирование:**
  + unit-тесты для сервисов Nest.js с помощью Jest.
* **Безопасность:**
  + Проверка и защита от уязвимостей: санитизация пользовательского ввода, защита от SQL-инъекций (Prisma делает это автоматически), корректная настройка CORS.
* **Производительность:**
  + Оптимизация React-компонентов (React.memo, useMemo, useCallback).
  + Оптимизация изображений.
  + Инструменты анализа: Lighthouse.

**Рекомендуемые ресурсы для обучения:**

1. **Документация:** Всегда читайте в первую очередь официальную документацию (nextjs.org, nestjs.com, prisma.io, tailwindcss.com). Она отличного качества.
2. **YouTube-каналы:**
   * **Владилен Минин** — отличные фундаментальные курсы по JS и React.
   * **Ulbi TV** — лучший контент по Next.js и Nest.js на русском языке.
3. **Курсы:**
   * **TheNetNinja** (YouTube, платные курсы) — качественные и структурированные курсы по всему стеку.
4. **Книги:** "Вы не знаете JS" (Kyle Simpson) — для глубокого понимания JavaScript.

**Стратегия обучения:**

* **Не просто смотреть, а делать.** После каждого урока обязательно пишите код самостоятельно.
* **Дробите большие задачи на маленькие шаги.** Не «сделать аутентификацию», а «1. создать форму входа, 2. отправить запрос на сервер, 3. обработать ответ...».
* **Гуглите ошибки.** Умение читать и понимать сообщения об ошибках — ключевой навык разработчика.
* **Сразу начинайте вести код в Git.**

Этот план — ваш план-график на ближайшие полгода-год. Следуя ему, вы не просто изучите технологии, вы создадите реальный, сложный и качественный продукт, который будет отличным пунктом в вашем портфолио. Удачи! Это захватывающее путешествие.