

Возможности и ограничения больших языковых моделей при генерации веб-приложений в парадигме Zero-Dependency на примере шахматного аналитического движка

Пайдершев Алексей Константинович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье исследуются возможности и инженерные ограничения больших языковых моделей при автономной генерации программного кода в архитектурной парадигме Zero-Dependency. С использованием методологии детализированного промпт-инжиниринга осуществлена поэтапная разработка комплексного веб-приложения – интерактивной шахматной доски с аналитическим движком, реализованной в едином файле (HTML, CSS и Vanilla JavaScript). Результаты эксперимента демонстрируют высокую эффективность искусственного интеллекта в проектировании реактивных пользовательских интерфейсов, программировании строгих математических конечных автоматов и оптимизации структур данных. Сделан вывод о том, что генеративная разработка позволяет оперативно создавать легковесные и независимые веб-программы для работы со строгой логикой и структурированными данными, однако требует перехода к гибридным архитектурам при столкновении с задачами семантического анализа медиа.

Ключевые слова: Qwen Studio, нейросети, промпт-инжиниринг, веб-разработка, вайб-кодинг, Zero-Dependency.

Capabilities and limitations of large language models in generating web applications in the Zero-Dependency paradigm using the example of a chess analytics engine

Paydershev Alexey Konstantinovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article explores the capabilities and engineering limitations of large language models in the context of autonomous generation of program code in the Zero-Dependency architectural paradigm. Using the methodology of detailed reverse engineering, a comprehensive web application, an interactive chessboard with an analytical engine implemented in a single file (HTML, CSS, and Vanilla JavaScript), was developed in a step-by-step manner. The results of the experiment demonstrate the high efficiency of artificial intelligence in designing reactive user interfaces, programming strict mathematical finite state machines, and optimizing data

structures. It is concluded that generative development allows for the rapid creation of lightweight and independent web programs for working with strict logic and structured data, but requires the transition to hybrid architectures when faced with the challenges of semantic media analysis.

Keywords: Qwen Studio, neural networks, prompt-engineering, web development, viber coding, Zero-Dependency.

1 Введение

1.1 Актуальность

В последние годы генеративные языковые модели и методы промпт-инжиниринга кардинально меняют процесс создания программного обеспечения, порождая новые подходы, такие как «вайб-кодинг». При этом современная веб-разработка часто страдает от избыточной зависимости от тяжеловесных фреймворков и внешних библиотек, что приводит к перегрузке приложений. В связи с этим возникает острая научная и практическая необходимость выяснить, способен ли искусственный интеллект выступать автономным системным архитектором и создавать сложные, оптимизированные веб-программы с нуля (в парадигме Zero-Dependency), опираясь исключительно на нативные технологии браузера – HTML, CSS и чистый JavaScript.

Кроме того, актуальность работы обусловлена потребностью провести четкую границу между маркетинговыми ожиданиями от нейросетей и инженерной реальностью. Разработчикам критически важно понимать, в каких задачах генеративный подход безупречен, а в каких – архитектурно несостоятелен. Разработка аналитического шахматного движка выступает в данном случае идеальным экспериментальным полигоном, результаты тестирования на котором позволят IT-специалистам избегать тупиковых ветвей разработки и применять ИИ максимально эффективно.

1.2 Обзор исследования

В статье С. В. Рогатина и В. А. Антошкина рассматриваются специфика и преимущества создания браузерных компьютерных игр на базе нативной технологии HTML5. Эта работа закладывает фундаментальное понимание базовых принципов разработки кроссплатформенных интерактивных приложений без использования сторонних плагинов [1]. А. Б. Рахмонов анализирует концепцию вайб-кодинга в контексте цифровой педагогики, оценивая его функциональные возможности для современного образовательного процесса [2]. Исследование И. В. Елманова посвящено инженерному контролю за ИИ, где для проверки сгенерированного методом vibe coding кода предлагается использовать методологию Spec Driven Development (разработка на основе спецификаций). Этот труд подтверждает критическую необходимость строгого архитектурного тестирования программ, созданных большими языковыми моделями [3]. Д. Р. Мухаметов расширяет понимание вайб-ориентированных ИИ-подходов, изучая их применение в «политическом вайб-маркетинге» для продвижения

общественных идей и конструирования образа будущего. Работа иллюстрирует гибкость генеративных нейросетей и их способность решать комплексные задачи далеко за пределами классического написания кода [4]. Фундаментальный обзор А. Д. Степанова, Л. А. Богданова, Д. К. Шишковой и А. Г. Кутихина демонстрирует практическое применение алгоритмических ИИ-инструментов в высокоточных научно-медицинских сферах, таких как трансляционная кардиология. Данная работа подтверждает глобальный тренд на успешное внедрение машинного интеллекта в дисциплины, требующие безошибочной логики и глубокого анализа структурированных данных [5].

1.3 Цель исследования

Цель исследования заключается в оценке эффективности и определении границ применимости больших языковых моделей (на примере Qwen Studio) при автономной генерации сложных, управляемых данными веб-приложений в рамках строгой архитектурной парадигмы Zero-Dependency (полный отказ от сторонних библиотек и фреймворков).

2 Материалы и методы

В основе данного исследования лежит подход, при котором традиционное ручное программирование заменяется генерацией кода с помощью больших языковых моделей (на примере Qwen Studio) без привлечения сторонних библиотек (Zero-Dependency). Основным инструментом выступает детальный промпт-инжиниринг – процесс составления точных текстовых запросов, которые переводят требования к алгоритмам и дизайну в рабочий программный код.

В качестве тестового проекта выбрана разработка интерактивной шахматной доски с функцией анализа партий. Этот пример идеально подходит для эксперимента, так как требует от нейросети объединить сложную математику, строгие игровые правила и отзывчивый интерфейс.

3 Результаты

Ключевым условием эксперимента стал полный отказ от внешних фреймворков и готовых пакетов для создания шахмат. Всё приложение размещается в одном файле, который содержит только базовый HTML, CSS и чистый JavaScript. Такое ограничение помогает выяснить, может ли искусственный интеллект самостоятельно выстраивать структуру веб-страницы и управлять ею, грамотно использовать встроенные функции браузера (например, захват и перетаскивание фигур мышью) и создавать оптимизированную программу, которая работает быстро и без сбоев без помощи сторонних решений.

Чтобы нейросеть лучше справилась с задачей, исходный запрос разделен на три независимых блока:

Интерфейс и дизайн: Модели заданы строгие правила визуального оформления. Например, ИИ должен использовать тени для визуального отделения черных фигур от темных клеток и применять плавные анимации для шкалы оценки.

Все графические элементы создавались исключительно с помощью текстовых символов и стилей, без загрузки картинок.

гровая логика: Нейросети поручено запрограммировать строгие шахматные правила. От алгоритма требовалось распознавать допустимые ходы, учитывать особые действия (рокировка, взятие на проходе) и блокировать ходы, подставляющие короля под удар.

налитический модуль: ИИ получил задачу написать функцию, которая мгновенно оценивает позицию на доске после каждого хода. Она должна учитывать стоимость фигур и выгодность их расположения (например, контроль центра), выдавая в итоге числовую оценку ситуации.

Особое внимание уделялось тому, как ИИ связывает внутренние математические расчеты с картинкой на экране, используя только стандартный JavaScript. Приложение должно работать по принципу реактивности: любое перемещение фигуры в коде должно немедленно менять интерфейс. Нейросети предстояло реализовать динамический подсчет и вывод значков «съеденных» фигур, плавное движение ползунка на шкале оценки в зависимости от того, кто выигрывает и автоматическую запись истории ходов в классической шахматной нотации с генерацией соответствующих текстовых блоков.

Промт для генерации кода: «Действуй как Senior Frontend разработчик. Твоя задача – написать полный, готовый к запуску код веб-приложения «Интерактивная шахматная доска с аналитикой и историей ходов». Приложение должно быть реализовано в одном файле (HTML, CSS и JavaScript) и работать локально в браузере. Категорически запрещено использовать сторонние библиотеки и изображения (все иконки и фигуры – через CSS и Unicode).

Архитектурные требования (UI/UX):

1. Общий дизайн: современный, эстетичный интерфейс (в стиле Lichess или Chess.com). Используй мягкие тени, скругленные углы и приятную темную тему для фона приложения. Шахматная доска должна иметь элегантную палитру.

2. Отрисовка фигур: используй стандартные Unicode-символы.

- Важно: белые фигуры должны быть строго белого цвета, а черные – строго черного. Чтобы они не сливались с клетками доски, обязательно добавь им контрастную обводку (например, белая тень для черных фигур и черная тень для белых).

- Реализуй плавное перетаскивание фигур.

3. Эстетичная индикация ходов: при клике на фигуру подсвечивай возможные ходы красиво. Вместо заливки всей клетки используй:

- Для пустых клеток: аккуратный полупрозрачный круг по центру клетки.

- Для клеток с фигурами противника: контрастное кольцо или рамку по краям клетки.

4. Шкала оценки позиции: вертикальная полоса слева от доски.

- Она должна заполняться белым и черным цветами.

- Добавь плавное и мягкое движение границы при изменении оценки.
- Выводи точное числовое значение прямо внутри шкалы или рядом с ней.

5. Панель «История ходов» и Статус: справа от доски размести стильный блок с историей партии. Запись ходов должна вестись в алгебраической нотации с использованием Unicode-значков фигур. Под историей выводи текстовый статус игры (чей ход, шах, мат, пат).

6. Индикатор материала: над доской (для черных) и под доской (для белых) выводи аккуратные мини-иконки съеденных фигур. Если есть материальный перевес, рядом с иконками должен появляться четкий числовой индикатор (например, «+1», «+3»).

Требования к игровой логике:

- Режим «Песочницы»: пользователь делает ходы за обе стороны. ИИ-соперника нет.

- Строгая валидация всех ходов (рокировка, взятие на проходе, превращение пешки в ферзя).

- Запрет на ходы, оставляющие короля под шахом. Проверка на мат и пат.

Требования к Аналитическому движку:

- Напиши синхронную функцию статической оценки позиции, запускаемую после каждого хода.

- Учитывай материальный баланс (Пешка=1, Конь=3, Слон=3, Ладья=5, Ферзь=9) и базовые позиционные бонусы (контроль центра).

- Возвращай итоговый счёт, который немедленно обновляет плавнодвигающуюся шкалу оценки и пересчитывает цифры материального перевеса в интерфейсе.

Код должен быть чистым, хорошо структурированным, с комментариями на русском языке. Выведи полный код в одном блоке».

Результат: <https://drive.google.com/file/d/1Z5N54v6nu1MAinYYnM7-xk1GyV1HW6qz/view?usp=sharing>

В результате эксперимента с помощью точных текстовых запросов к языковой модели Qwen Studio удалось создать рабочий прототип шахматной доски с функцией анализа партии (рис. 1).

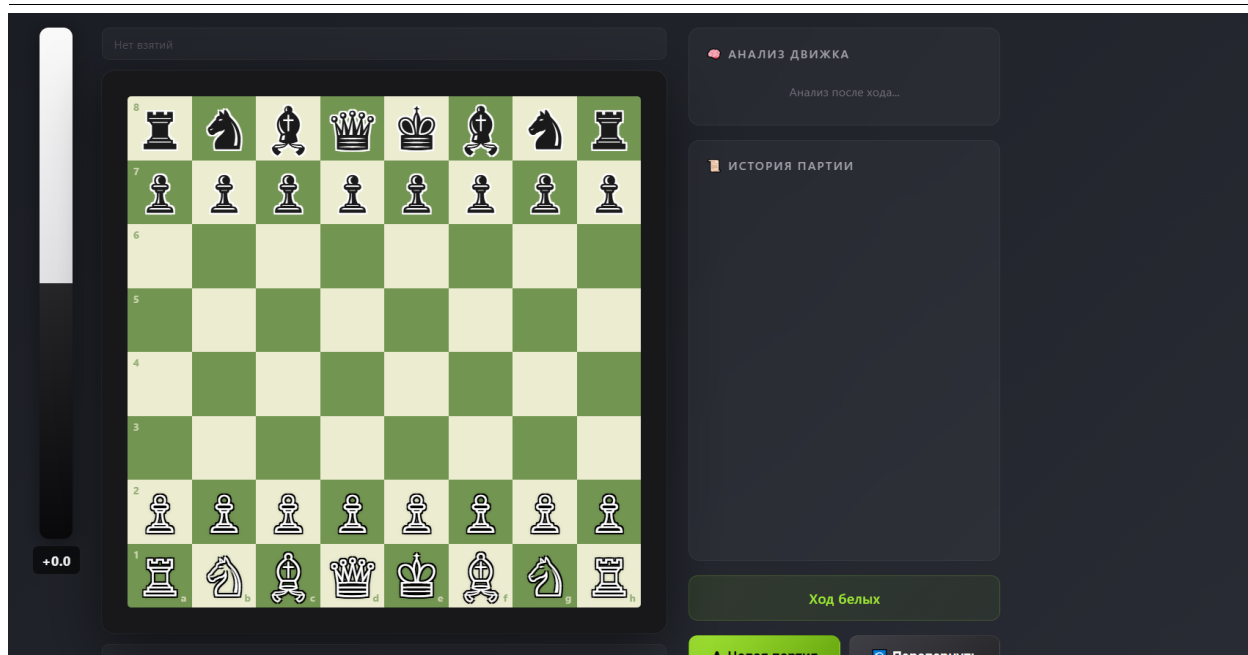


Рисунок 1. Интерфейс приложения

Языковая модель строго соблюла ограничения, спроектировав приложение в рамках единого файла. Интерфейс успешно реализован средствами нативного HTML и CSS. Нейросеть продемонстрировала высокую точность в управлении визуальным рендерингом.

Проблема слияния белых фигур со светлыми клетками доски решена с помощью добавления теней, что позволило добиться контрастности без загрузки внешних картинок.

Подсветка возможных ходов реализована аккуратно: программа генерирует полупрозрачные точки и рамки, делая внешний вид приложения эстетичным и похожим на современные коммерческие продукты. Перетаскивание фигур мышью работает плавно и без сбоев благодаря правильному использованию встроенных функций браузера (рис. 2).

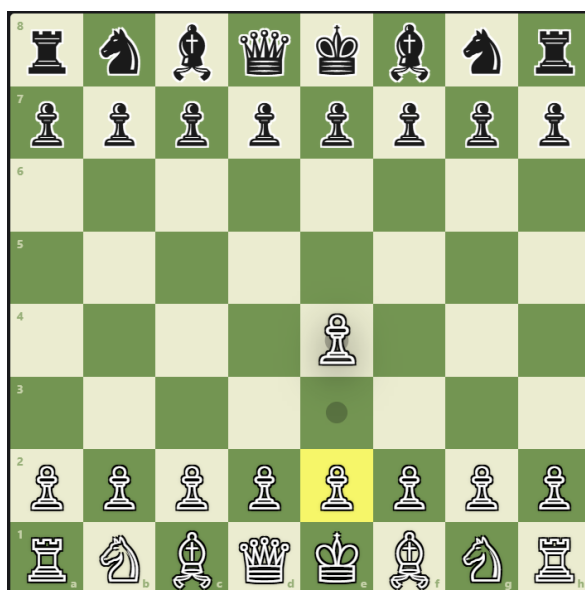


Рисунок 2. Перетаскивание фигур

Сгенерированный код безошибочно воспроизводит сложные шахматные правила. Программа в реальном времени определяет допустимые ходы, корректно обрабатывая даже такие специфические ситуации, как взятие на проходе, рокировка и превращение пешки. Механизм проверки шаха, мата и пата работает синхронно, мгновенно запрещая пользователю делать неверные ходы. Это доказывает, что искусственный интеллект способен выстраивать строгую и надежную логику работы приложения.

Самым важным достижением стала способность приложения мгновенно обновлять интерфейс в ответ на действия пользователя, используя исключительно базовый JavaScript:

аналитическая функция правильно переводит такие показатели, как ценность фигур и контроль центра доски, в итоговую числовую оценку.

Любое изменение этой оценки сразу отражается на визуальной шкале. Нейросеть применила CSS-анимации, чтобы ползунок двигался плавно, без резких скачков (рис. 3).

Кольцо истории партий автоматически переводит координаты клеток в стандартную шахматную нотацию, самостоятельно добавляя нужные значки фигур после каждого хода (рис. 4).

Отображение съеденных фигур работает верно, показывая съеденные фигуры и материальное преимущество для обеих сторон (рис. 5).



Рисунок 3. Визуальная шкала оценки

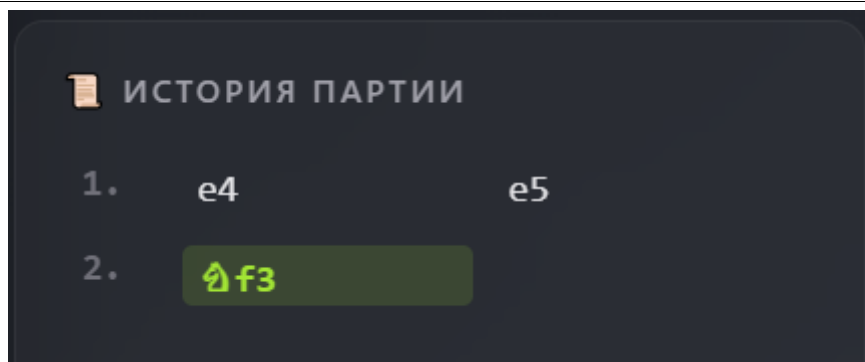


Рисунок 4. История партии

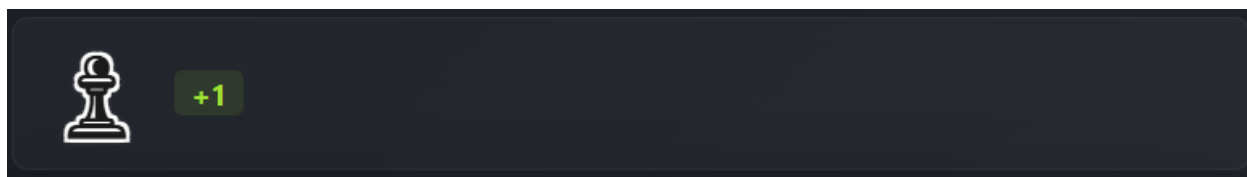


Рисунок 5. Отображение материального преимущества

Также в приложении реализована функция переворота доски для удобства анализа партии как при игре за белых, так и за чёрных.

Функция анализа движка, которая должна показывать лучшие ходы в сложившейся партии, оказалась не рабочей. Данная проблема решена посредством написания нового текстового запроса для нейросети.

Промт: «Функция анализ движка не работает. С чем это может быть связано и как это исправить?»

Р

В ответ на запрос нейросеть не просто объяснила, как исправить данную проблему, а предоставила полностью исправленный функциональный код, в котором решена проблема с анализом движка (рис. 6).

Л
Б
Т
а
Т

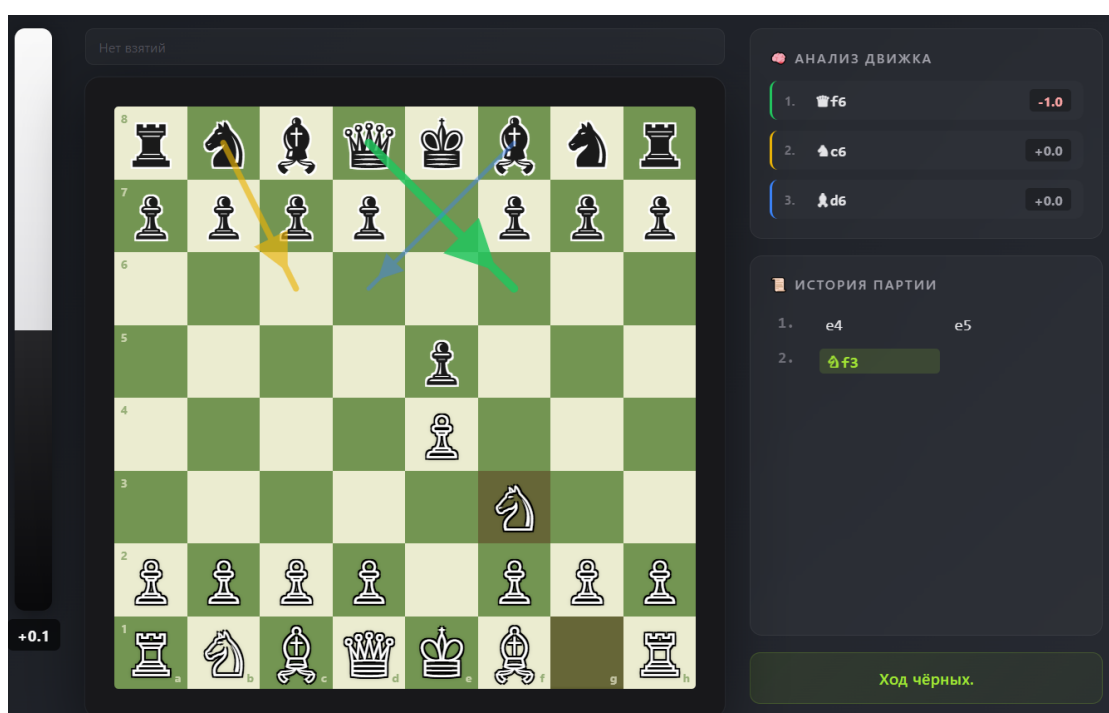


Рисунок 6. Анализ движка

Данная функция оценивает позицию по материальному балансу, контролю центральных клеток, развитию лёгких фигур и предлагает на выбор топ-3 лучших хода. Зелёные стрелки указывают на самый сильный ход, жёлтые на второй и синие на третий по силе.

Функция анализа движка является по сути самой важной в подобных шахматных программах, так как позволяет не только анализировать текущую ситуацию на доске, но и обучаться.

Приложение соответствует всем шахматным правилам и учитывает даже сложные ситуации (например, не позволяет сделать рокировку, если король находится под шахом или на его пути находятся поля под боем) (рис. 7).



Рисунок 7. Шах королю

Также реализована функция превращения пешки. Как правило, подобные движки автоматически превращают её в ферзя, но поскольку по правилам игрок сам может выбрать фигуру для превращения, приложение предлагает выбор (рис. 8).

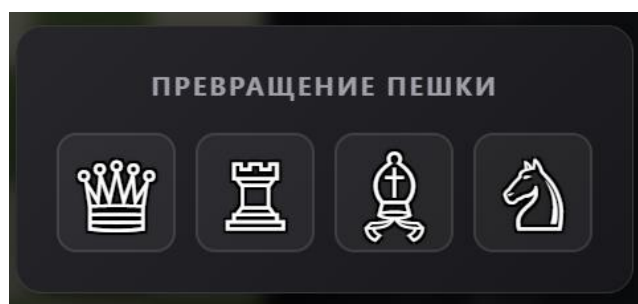


Рисунок 8. Превращение пешки

По завершении нейросеть сама предложила добавить в программу анализ дебюта и загрузку скриншотов уже сыгранных партий для разбора. Эта функция превращает приложение из строго аналитического с элементами обучения в полностью обучающее.

Данные функции напрямую реализовать не получилось ввиду нескольких причин:

азовые средства браузера умеют работать с изображениями только как с матрицей пикселей. Сама программа не понимает, что группа пикселей образует «белого коня» или «черную пешку». Чтобы научить алгоритм распознавать фигуры, необходимо с нуля запрограммировать сложную систему компьютерного зрения, которая будет сопоставлять контуры на картинке с заложенными в код шаблонам.

олноценные базы дебютов весят слишком много. Если попытаться уместить их целиком в один JavaScript-файл в виде огромного массива, браузер просто зависнет при загрузке страницы или исчерпает лимит памяти.

Первую проблему удалось решить. Загрузка скриншотов заменена поддержкой стандартных шахматных нотаций – PGN, которые доступны по окончании партии на Lichess и Chess.com. Нейросети поставлена задача написать алгоритм, который считывает текстовую строку, расшифровывает ее и расставляет фигуры на доске.

Промт: «Замени загрузку скриншота на загрузку PGN-нотации».

Результат:

<https://drive.google.com/file/d/16PyrZilpBfwJzeHtThJVzTTnEkFkUT5i/view?usp=sharing>

Нейросеть успешно справилась с поставленной задачей и переписала код, который стал поддерживать PGN-нотации (Рис. 9-10). При загрузке PGN на доске появляется уже отыгранная партия (рис. 11).

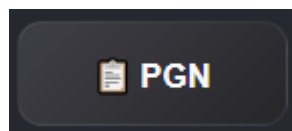


Рисунок 9. Кнопка импорта

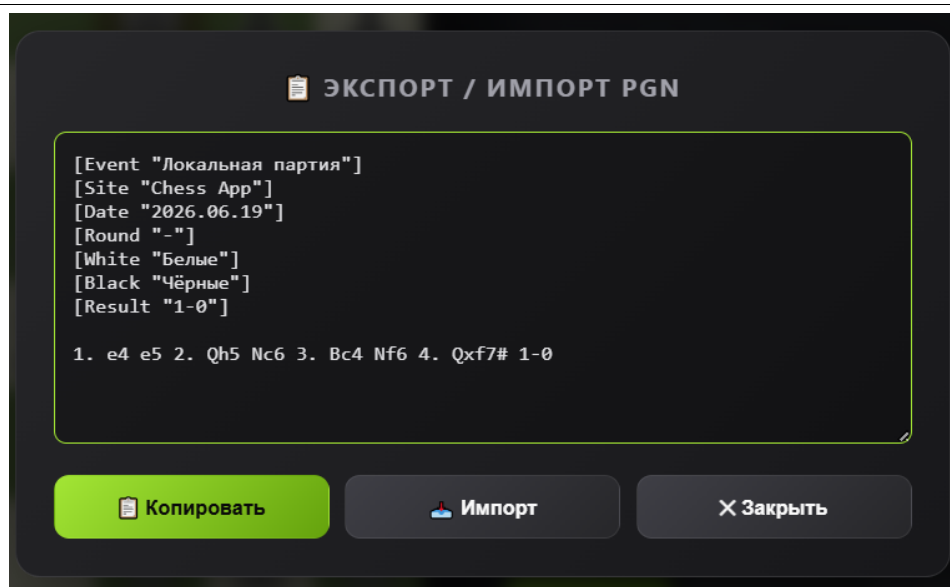


Рисунок 10. Импорт PGN-нотации



Рисунок 11. Импортированная партия

Проблема с анализом дебюта оказалась сложнее. Как указано ранее, для её исправления необходимо уместить в коде слишком большое количество информации. В таком случае, даже если получится написать такой код с помощью нейросети (что тоже трудно реализуемо, так как модели с таким большим контекстным окном требуют больших ресурсов и стоят достаточно дорого), браузер просто сломается при попытке загрузить приложение. Чтобы реализовать данную функцию необходимо подключать сторонние библиотеки, что не соответствует логике исследования.

Для того, чтобы реализовать эту функцию в рамках одного кода, необходимо спроектировать компактную «хэш-таблицу», которая будет содержать текстовую строку текущей позиции и базовые шахматные дебюты.

Промт: «Возникла проблема с распознаванием дебюта. Давай будем использовать легковесный подход. Спроектируй компактную структуру данных – «хэш-таблицу» (словарь). В ней в качестве ключа используется текстовая строка текущей позиции (PGN), а в качестве значения – название дебюта (например, «Испанская партия» или «Защита Каро-Канн»). Перепиши код».

Результат:

В результате получилось реализовать функцию анализа дебюта (рис. 12).



Рисунок 12. Анализ дебюта

Такой легковесный подход обладает несколькими преимуществами перед большим массивом данных:

компактность – всего 50 записей содержат основные дебюты

высокая скорость – быстрый поиск по маленькой таблице

надёжность – нет сложной логики, только точное совпадение строки

легко расширять – написание одной строки добавляет один новый дебют

Теперь после каждого хода алгоритм мгновенно проверяет совпадение позиции со словарем. Это решение не требует сложных вычислений, работает без задержек и не перегружает процессор. Эксперимент показал, что языковая модель способна находить элегантные архитектурные компромиссы, заменяя тяжелые базы данных на быстрые и легковесные словари, идеально подходящие для веб-приложений.

4 Вывод

Подводя итоги проведенного исследования, можно сделать вывод, что использование больших языковых моделей в связке с детализированным промпт-инжинирингом представляет собой высокоэффективный метод разработки сложных веб-приложений. Эксперимент по созданию интерактивной шахматной доски с аналитическим движком без использования сторонних библиотек подтвердил ряд важных гипотез и выявил четкие границы применимости данного подхода.

Исследование показало, что генеративные модели способны выходить за рамки написания разрозненных функций. Нейросеть успешно спроектировала архитектуру приложения в едином файле, связав сложную математическую логику (правила шахмат, статический анализ) с пользовательским интерфейсом. Решение таких задач, как создание реактивной шкалы оценки и плавная анимация фигур средствами чистого JavaScript и CSS, доказывает способность ИИ генерировать оптимизированный и эстетичный код, сопоставимый с коммерческими продуктами.

Успешная реализация функции анализа дебютов и загрузки партий стала одним из ключевых достижений эксперимента. Она продемонстрировала способность языковой модели к алгоритмической оптимизации. Вместо интеграции тяжелых баз данных, ИИ использовал компактные структуры (хэш-таблицы), что позволило приложению мгновенно распознавать дебюты и анализировать позиции, сохраняя минимальный вес файла и высокую скорость работы в браузере.

Попытка внедрения функции распознавания позиций по скриншоту выявила фундаментальное ограничение подхода. Разработка систем компьютерного зрения с нуля без сторонних библиотек оказалась нецелесообразной и привела к нестабильности кода. Этот отрицательный результат позволяет сформулировать важное правило: генеративный подход в парадигме Zero-Dependency идеален для работы со строгой логикой и структурированными форматами, но неприменим для семантического анализа неструктурированных медиаданных.

Автоматизированная генерация кода с помощью ИИ фундаментально меняет процесс проектирования интерфейсов. Разработчик переходит от ручного написания строк кода к роли постановщика задач и верификатора архитектуры. Если задача опирается на строгие правила, математику и структурированные данные, искусственный интеллект позволяет в кратчайшие сроки создавать надежные, независимые от фреймворков и реактивные веб-приложения.

Библиографический список

1. Рогатин С. В., Антошкин В. А. Разработка компьютерных игр по технологии html5 //Информатика и прикладная математика. 2015. №. 21. С. 12-16. URL: <https://elibrary.ru/item.asp?id=25030492>

2. Рахмонов А. Б. Вайб-кодинг как инструмент цифровой педагогики: функциональные возможности и механизмы реализации //The Lingua Spectrum. 2025. Т. 12. №. 2. С. 552-556. URL: <https://lingvospektr.uz/index.php/Ingsp/article/view/1329>
3. Елманов И. В. Инженерный подход к ии: использование spec driven development для валидации vibe coding //Вестник науки. 2025. Т. 4. №. 12 (93). С. 716-723. URL: <https://cyberleninka.ru/article/n/inzhenernyy-podhod-k-ii-ispolzovanie-spec-driven-development-dlya-validatsii-vibe-coding>
4. Мухаметов Д. Р. Политический вайб-маркетинг: возможности искусственного интеллекта для продвижения образа будущего //Известия Тульского государственного университета. Гуманитарные науки. 2025. №. 3. С. 27-33. URL: <https://cyberleninka.ru/article/n/politicheskiy-vayb-marketing-vozmozhnosti-iskusstvennogo-intellekta-dlya-prodvizheniya-obraza-buduschego>
5. Степанов А. Д., Богданов Л. А., Шишкова Д. К., Кутихин А. Г. Применение инструментов на основе искусственного интеллекта в экспериментальной и трансляционной кардиологии: обзор //Российский кардиологический журнал. 2026. Т. 31. №. S2. С. 114-122. URL: <https://cyberleninka.ru/article/n/primenenie-instrumentov-na-osnove-iskusstvennogo-intellekta-v-eksperimentalnoy-i-translyatsionnoy-kardiologii-obzor>