Слайд 1: О себе

Студент магистратуры, окончил бакалавриат по направлению ИВТ в 2024 году.

Слайд 2: Написание кода в 2025 году

Рассмотрим три подхода к написанию кода в современных условиях: классическое программирование, вайбкодинг и метакодинг. Каждый подход имеет свои преимущества и недостатки.

Слайд 3: Классическое программирование

Написание кода без использования ИИ: разработчик самостоятельно решает задачи, изучает документацию и ищет решения.
Преимущества:

- Глубокое понимание алгоритмов, паттернов и архитектуры.
- Устойчивость к отсутствию доступа к ИИ.
- Полный контроль над кодом и понимание причин ошибок.
- Развитие профессиональных навыков и инженерного мышления.

Недостаток:

- Низкая скорость работы в сравнении с ИИ-подходами, что становится критичным в условиях конкуренции и оптимизации процессов.

Вывод:
Классическое программирование развивает надежных специалистов, но его ценность зависит от темпов работы, задач и требований рынка.

Слайд 4: Вайбкодинг

Вайбкодинг — подход, основанный на использовании ИИ для генерации кода без глубокого анализа результата. Появился с развитием языковых моделей (LLM).
Характеристики:

- Поверхностное использование ИИ без понимания предложенных решений.
- Минимальная проверка кода, копирование результатов.
- Отсутствие архитектурного подхода и документации.
- Зависимость от качества запросов к ИИ.

Преимущества:

- Высокая скорость написания кода.
- Возможность многозадачности.
- Простота взаимодействия через текстовые запросы.

Недостатки:

- Иллюзия компетентности: отсутствие понимания кода.
- Отсутствие профессионального роста из-за недостатка рефлексии.
- Сложности с поддержкой и качеством кода.

Вывод:
Вайбкодинг ускоряет выполнение задач, но ограничивает профессиональное развитие. ИИ эффективен только при осознанном использовании.

Слайд 5: Метакодинг

Метакодинг — осознанное использование ИИ как инструмента, усиливающего интеллект разработчика, а не заменяющего его.
Принципы:

- Контроль: Проверка и анализ кода, предложенного ИИ.
- Обратная связь: Запрос объяснений на доступном уровне.
- Чистый промтинг: Структурированные запросы для точных результатов.
- Ответственность: Качество результата зависит от постановки задачи.
- Цикл работы: Запрос → ответ → правка → тест → интеграция.

Практические советы:

- Просите ИИ объяснять термины и код.
- Читайте и проверяйте предложенные решения.
- Ведите документацию для контекста и минимизации ошибок.
- Переформулируйте запросы при ошибках, экономьте контекст.

Недостаток:
Требует дисциплины, чтобы не скатиться в вайбкодинг.
Вывод:
Метакодинг — инженерный подход, сочетающий человеческое мышление, дисциплину и возможности ИИ.

Слайд 6: Как использовать ИИ в разработке

ИИ — инструмент, повышающий продуктивность разработчика, но не заменяющий его.

Возможности ИИ:

- Объяснение кода и концепций, адаптация под уровень знаний.
- Код-ревью, поиск ошибок и оптимизация.
- Генерация идей, алгоритмов, архитектурных решений.
- Создание документации, тестов, коммитов.
- Ускорение рутинных задач и написания кода.
- Поддержка обучения через диалог и интерактивные задачи.
- Увеличение возможностей соло-разработчиков (например, создание MVP).

Вывод:
ИИ помогает быстрее решать задачи и учиться, если использовать его осознанно.

Слайд 7: Личный опыт. Миграция данных

Задача: Расширение модели данных с сохранением существующих данных.
Проблема: Предыдущий опыт миграции привел к потере данных.
Решение с ИИ: ИИ проанализировал модель, предложил правильные атрибуты и шаги миграции, объяснил процесс.
Результат: Успешное расширение модели, обучение правильной миграции, снижение стресса.

Слайд 8: Личный опыт. Оптимизация

Задача: Добавление аннотаций (lollipop) к графикам статистики упражнений.
Проблема: ИИ-решения вызывали проблемы с производительностью (двойной скролл, непредвиденное поведение).
Решение: ИИ помог оптимизировать код, устранив проблемы после анализа и переформулировки запросов.
Результат: Стабильная работа графиков, улучшение производительности.

Слайд 9: Личный опыт. Ошибки слепого копирования

Задача: Реализация функции вывода всех подходов упражнения за всё время.
Ошибка: Делегирование задачи ИИ без проверки, копирование кода

привело к низкой производительности.
Вывод: Неосознанное использование ИИ может привести к некачественным решениям. Проверка и анализ кода обязательны.

Слайд 10: Навыки программиста в 2025 году

Предлагается обсуждение: какие навыки необходимы современному разработчику?
Хардскиллы:

- Знание алгоритмов и структур данных.
- Понимание архитектуры ПО (MVC, MVVM и др.).
- Принципы ООП/функционального программирования, шаблоны проектирования.
- Работа с Git и системами контроля версий.
- Написание чистого, читаемого кода.
- Владение стеком технологий (например, JS + React, SwiftUI + SwiftData).
- Работа со сторонними библиотеками.
- Знание методологий разработки (Scrum, Agile).
- Навыки тестирования.

Софтскиллы:

- Умение работать в команде, давать и принимать фидбек.
- Владение английским для чтения документации.
- Грамотная коммуникация, проактивность.
- Способность учиться на ошибках.

Метанавыки:

- Самообучение и освоение новых технологий.
- Системное мышление и понимание бизнес-целей.
- Гибкость в работе с разными технологиями.
- Владение ИИ-инструментами и их интеграция в процессы.

Слайд 11: Мотивация

- Принцип кайдзен: Постоянное улучшение через небольшие шаги (30 минут кода, 10 минут английского, 1 коммит в день).
- Паралич выбора: Важно начать, а не выбирать идеальный путь. Практика важнее стратегии.

- Глобальные возможности: Системный подход, портфолио на GitHub, знание английского и участие в сообществах открывают перспективы удаленной работы и роста.
- Рекомендации: Пишите код ежедневно, ведите портфолио, делитесь знаниями в соцсетях, изучайте английский.

Слайд 12: Инструменты ИИ

- Cursor AI: ИИ-помощник в IDE (на базе VS Code). Поддерживает написание, рефакторинг, объяснение кода с учетом контекста проекта.
- Windsurf: Облачная альтернатива от создателей Replit. Работает в браузере, поддерживает сложные задачи и code agents.
- GitHub Copilot: Хорош для автодополнения, но ограничен в контексте проекта.
- Функции ИИ-инструментов:
    - Контекстные подсказки и автодополнение.
    - Генерация кода, исправление ошибок.
    - Поддержка запросов на естественном языке.
    - Создание документации и анализ кода.