Junior Full-Stack Developer with Operations Support — Тестовое задание

> **Позиция:** Junior Full-Stack Developer with Operations Support — Orbios LLC
> **Формат:** Part-time, 20 часов/неделю
> **Время выполнения:** 2-3 часа максимум
> **Дата создания:** 2025-12-10

📋 Общая информация

Цель задания: Оценить технические навыки разработки, использование AI-инструментов и способность поддерживать операционные процессы.

Формат: Выполните задание локально и отправьте результат через GitHub или Google Drive.

Важно:

- ✅ **Обязательно используйте AI-инструменты** (Claude Code, Cursor, ChatGPT) — это ключевое требование!
- ✅ **Обязательно используйте хотя бы один MCP Server** — для интеграции с внешними системами или данными (репозиторий, веб, память и т.д.)
- ✅ Фокус на качестве, а не на сложности
- ✅ Документируйте процесс работы с AI

🎯 Задание

Ситуация

Вы работаете Junior Full-Stack Developer в распределенной команде. Вам нужно:

1. Создать простое веб-приложение
2. Документировать процесс разработки
3. Создать инструкцию для команды

💻 Часть 1: Разработка (60-90 минут)

Задача

Создайте простое **Task Tracker** приложение с использованием React и TypeScript.

Требования:

Технологии:

- React + TypeScript
- Можно использовать Next.js (опционально, но приветствуется)
- Стилизация на ваш выбор (CSS, Tailwind, или простые стили)

Функционал:

1. **Список задач**

   - Отображение списка задач
   - Каждая задача имеет: название, описание, статус (To Do / In Progress / Done)
   - Задачи можно добавлять, редактировать, удалять

2. **Базовые операции**

   - Добавление новой задачи
   - Изменение статуса задачи (перетаскивание или кнопки)
   - Редактирование задачи
   - Удаление задачи

3. **Простое хранение данных**
   - Можно использовать localStorage для сохранения данных
   - Или создать простой JSON файл (если используете Next.js)

Что НЕ требуется:

- ❌ Сложная архитектура
- ❌ Backend сервер
- ❌ База данных
- ❌ Аутентификация
- ❌ Сложная стилизация

Что важно:

- ✅ Работающий код
- ✅ Чистый и читаемый код
- ✅ TypeScript типизация
- ✅ Использование AI-инструментов для помощи

Критерии оценки:

- ✅ Приложение работает без ошибок
- ✅ Код структурирован и читаем
- ✅ Использованы TypeScript типы
- ✅ Базовый UI/UX (приятно смотреть и использовать)

🤖 Часть 2: Документация AI-процесса (20-30 минут)

Задача

Создайте документ, описывающий **как вы использовали AI-инструменты** при разработке.

Требования:

Структура документа:

1. **Какой AI-инструмент использовали?**
   - Claude Code / Cursor / ChatGPT / другой
   - Почему выбрали именно его?

2. **Какие задачи решали с помощью AI?**
   - Примеры: генерация кода, исправление ошибок, объяснение концепций
   - Приведите 2-3 конкретных примера

3. **Примеры промптов**
   - Покажите 2-3 промпта, которые вы использовали
   - Опишите, что получили в ответе

4. **Что вы узнали?**
   - Какие новые подходы или техники узнали благодаря AI?
   - Как AI помог ускорить разработку?

5. **Трудности и решения**
   - Были ли проблемы при работе с AI?
   - Как их решили?

Формат:

- Markdown файл или Google Docs
- Название: `AI_WORKFLOW_DOCUMENTATION.md`

Критерии оценки:

- ✅ Понятное описание процесса
- ✅ Конкретные примеры использования AI
- ✅ Рефлексия о процессе
- ✅ Показывает эффективное использование AI-инструментов

📚 Часть 3: Операционная документация (30-40 минут)

Задача

Создайте **инструкцию для команды** о том, как использовать ваше Task Tracker приложение.

Требования:

Структура инструкции:

1. **Краткое описание**
   - Что это за приложение
   - Для чего оно нужно
2. **Как установить и запустить**
   - Пошаговая инструкция
   - Требования (Node.js версия, зависимости)
   - Команды для запуска
3. **Как использовать**
   - Как добавить задачу
   - Как изменить статус
   - Как редактировать/удалить задачу
4. **FAQ (3-5 вопросов)**
   - Часто задаваемые вопросы и ответы
   - Например: "Где хранятся данные?", "Как сбросить все задачи?"
5. **Контакты для помощи**
   - К кому обращаться при проблемах

Формат:

- Markdown файл: `TASK_TRACKER_INSTRUCTIONS.md`
- Или можно создать в README.md проекта
- Понятный, структурированный текст
- Можно добавить скриншоты (опционально)

Критерии оценки:

- ✅ Полнота информации
- ✅ Понятность для не-технических пользователей
- ✅ Структурированность
- ✅ Практичность инструкций

🔧 Часть 4: MCP Integration (30-40 минут)

Задача

Обязательное требование: в решении должен быть использован хотя бы один MCP Server.

Возможные варианты MCP Server-ов

Ниже несколько примеров MCP, которые вы можете использовать (достаточно выбрать один):

- **GitHub MCP Server**
  Использование репозитория как источника правды: просмотр и поиск кода, анализ структуры проекта, работа с issues и pull requests.

- **Bright Data MCP**
  Получение данных с веб-страниц и поисковых систем: извлечение содержимого страниц, сбор актуальной информации для документации или аналитики.

- **Context7 MCP**
  Доступ к внешним базам знаний и документам: расширение контекста при работе над задачами, поиск информации в документации.

- **Memory MCP**
  Хранение и использование долговременной памяти агента: предпочтения пользователя, состояние задач, важные контексты между сессиями.

Вы можете выбрать любой MCP Server, который лучше всего подходит к вашему рабочему процессу и стеку. Главное — **показать, что вы умеете интегрировать MCP в процесс разработки или операционную часть** (например, помощь с репозиторием, документацией, данными или контекстом).

Что показать в результате

- Какой MCP Server вы выбрали
- Для чего вы его использовали (1–2 конкретных сценария)
- Как это помогает в разработке или операционной работе с проектом

🚀 Часть 5: Создание собственного MCP Server (Опционально, +30-40 минут)

Задача

Опциональное задание: Создайте свой собственный MCP Server для автоматизации какой-либо задачи в вашем проекте.

Требования:

Примеры MCP серверов, которые можно создать:

1. **Task Tracker MCP Server**
   - Автоматизация управления задачами через MCP
   - Интеграция с вашим Task Tracker приложением
   - Примеры: создание задач, обновление статусов, генерация отчетов

2. **Code Quality MCP Server**
   - Автоматический анализ качества кода
   - Проверка соответствия стандартам кодирования
   - Генерация метрик и отчетов

3. **Documentation MCP Server**
   - Автоматическая генерация документации
   - Обновление README и других документов
   - Синхронизация документации с кодом

4. **Deployment MCP Server**
   - Автоматизация деплоя приложения
   - Управление окружениями
   - Мониторинг статуса деплоя

Что нужно сделать:

1. **Создайте MCP Server**
   - Выберите задачу для автоматизации
   - Реализуйте MCP Server с использованием TypeScript/JavaScript
   - Добавьте минимум 2-3 инструмента (tools)

2. **Интегрируйте с проектом**
   - Подключите MCP Server к вашему Task Tracker
   - Покажите примеры использования

3. **Документируйте**
   - Опишите архитектуру MCP Server
   - Покажите примеры кода
   - Объясните, какую проблему решает ваш сервер

Формат:

- Код MCP Server в папке `mcp-server/` или отдельном репозитории
- Документация в `MCP_SERVER_DOCUMENTATION.md`
- Примеры использования и конфигурации

Критерии оценки:

- ✅ MCP Server работает и интегрирован с проектом
- ✅ Код чистый и хорошо структурирован
- ✅ Решает реальную задачу автоматизации

- ✅ Хорошо документирован

Примечание: Это опциональное задание. Если вы его выполните, это будет дополнительным плюсом при оценке.

📝 Часть 6: Описание процесса разработки (15-20 минут)

Задача

Напишите краткое описание **вашего процесса разработки** для этого задания.

Вопросы для ответа:

1. **С чего вы начали?**
   - Какой был первый шаг?
   - Как планировали работу?

2. **Как организовали код?**
   - Какая структура файлов?
   - Почему выбрали такой подход?

3. **Какие были трудности?**
   - С чем столкнулись?
   - Как решали проблемы?

4. **Что можно улучшить?**
   - Что бы вы добавили, если было бы больше времени?
   - Какие функции расширили бы?

5. **Опыт работы с MCP**
   - Какой MCP Server вы использовали и как он помог в разработке?
   - Что было сложного при интеграции?
   - Если создали свой MCP Server — опишите процесс

Формат:

- Отдельный файл: `DEVELOPMENT_PROCESS.md`
- Или раздел в README
- Длина: 300-400 слов

📤 Как отправить результат

Вариант 1: GitHub (предпочтительно)

1. Создайте публичный репозиторий на GitHub
2. Загрузите код приложения
3. Добавьте файлы документации:
   - `AI_WORKFLOW_DOCUMENTATION.md`
   - `TASK_TRACKER_INSTRUCTIONS.md` (или в README.md)
   - `MCP_INTEGRATION.md` (документация по использованию MCP Server)
   - `DEVELOPMENT_PROCESS.md`
   - `MCP_SERVER_DOCUMENTATION.md` (если создали свой MCP Server)
4. Отправьте ссылку на репозиторий

Git-ветвление (опционально): приветствуется использование простой схемы веток вроде `main` / `dev` / `feature/*` с небольшими, осмысленными коммитами.

❓ Часто задаваемые вопросы

Q: Можно ли использовать готовые библиотеки/компоненты?
A: Да, можно использовать UI-библиотеки (например, Material-UI, Ant Design), но лучше показать свои навыки с базовым React.

Q: Нужен ли красивый дизайн?
A: Не обязательно, но приветствуется. Главное — функциональность и чистота кода.

Q: Что если не успеваю за 3 часа?
A: Лучше сделать меньше, но качественно. Опишите, что бы вы добавили при наличии времени.

Q: Можно ли использовать ChatGPT вместо Claude Code?
A: Да, можно использовать любой AI-инструмент. Главное — показать эффективное использование.

Q: Нужен ли backend?
A: Нет, достаточно localStorage или простого JSON файла.

Q: Что такое MCP Server и какой использовать?
A: MCP (Model Context Protocol) Server — это инструмент для интеграции AI-ассистентов с различными сервисами. **Обязательно используйте хотя бы один MCP Server** (например, GitHub MCP Server) для управления репозиторием, создания issues/PR, мониторинга CI/CD. Документация GitHub MCP: [GitHub MCP Server](https://github.blog/changelog/2025-09-04-remote-github-mcp-server-is-now-generally-available/)

Q: Обязательно ли создавать свой MCP Server?
A: Нет, это опциональное задание. **Обязательно только использование хотя бы одного MCP Server** (например, GitHub MCP Server). Создание своего MCP Server — дополнительный плюс и показывает продвинутые навыки.

📅 Сроки

Срок выполнения: 3 рабочих дня с момента получения задания

Если есть вопросы — не стесняйтесь писать! Мы готовы помочь.

Удачи! Мы с нетерпением ждем ваших результатов! 🚀