Вопросы работодателю на проект

pdf

Для занятых

Rus

1. какие темы/опыт в стеке js/angular более всего интересуют интервьюеров и проект? Реактивные формы, внедрение зависимостей, HTTP, Websocket, RxJS, шаблоны проектирования, тестирование, анимация, библиотека компонентов, вёрстка, lazy loading, дизайн система, соглашения по коду, JS/ES, асинхронный код/event loop, алгоритмы, typescript?
2. рабочее время по MSK TZ?
3. график и продолжительность обязательных совещаний?
4. Какие технологии используются для доступа к средам разработки: DOCKER, VPN, RDP, WEB, AZURE, AMAZON, VDI?
5. как часто случаются переносы сроков выполнения задач, как к ним относится Заказчик?
6. как давно развивается проект в котором я буду участвовать?
7. как оцениваете технический долг, как часто его оцениваете, как к нему относится Заказчик
8. есть ли правила улучшения кода - соглашения, линтеры
9. принято ли исправлять ошибки(рефакторить, подготавливать код) перед тем как реализовывать новый функционал
10. версии фреймворка, библиотек?
11. Время сборки проекта? CI/локально
12. на сколько спринтов вперёд готова аналитика/дизайн?
13. когда наступает/заканчивается новая/текущая веха?
14. что буду делать из перечисленного: рефакторинг, проектирование, документирование, программирование, тестирование, вёрстка, дизайн, devops(webpack/gulp/docker), администрирование(linux/nginx)
15. с кем я буду взаимодействовать в команде разработки? роль, опыт текущей роли(лет)?
16. есть ли в проекте внешние подрядчики или другие команды, с которыми я буду взаимодействовать и по каким вопросам?
17. кто в команде работает удалённо(не в офисе)?
18. мои дальнейшие действия: интервью, тестовое задание, проверка СБ, подписание соглашений?
19. какой порядок выхода из проекта, есть ли ограничения на минимальные сроки - день, неделя, месяц?

Eng

1. What topics/experiences in the JS/Angular stack are of the most interest to interviewers and the project? Reactive forms, dependency injection, HTTP, Websocket, RxJS, design patterns, testing, animation, component library, layout, lazy loading, design system, code conventions, JS/ES, asynchronous code/event loop, algorithms, typescript?
2. Working time in MSK TZ?
3. Schedule and duration of mandatory meetings?

4. What technologies are used to access the development environments: VPN, RDP, WEB, AZURE, AMAZON, VDI?
5. How often overtime work occur, how does the Customer react on them?
6. How long has the project been developing in which I will participate?
7. Versions of the framework, libraries?
8. Project build time? CI/local.
9. How many sprints ahead the analytics/design is ready?
10. When comes/ends a new/current milestone?
11. What will I do from the following: refactoring, design, documentation, programming, testing, layout, design, devops (webpack/gulp/docker), administration (linux/nginx)?
12. Who will I interact with in the development team? role, experience of the current role (years)
13. Is there any external contractors or other teams in the project with whom I will interact and on what issues
14. Who works remotely in the team (not in the office)?
15. My next steps: interview, test task, security check, signing agreements?
16. What is the procedure for exiting the project, are there any restrictions on the minimum terms - a day, a week, a month?

интервью

1. сколько времени на интервью у каждого участника
2. готовы ли сообщить мне обратную связь в конце технического интервью(что порадовало, что следует подтянуть)
3. мои дальнейшие действия: интервью, тестовое, проверка СБ, подписание соглашений?

организация

1. рабочее время по МСК
2. как ведётся график отпусков, можно ли согласовывать отпуска за полгода
3. какие правила/ограничения согласования внеплановых выходных(dayoff), больничных, отпусков
4. есть ли в проекте внешние подрядчики или другие команды, с которыми я буду взаимодействовать и по каким вопросам
5. график обязательных совещаний
6. как часто случаются переработки, как к ним относится Заказчик
7. что категорически нельзя делать разработчику
8. каковы ваши ожидания, опишите идеального кандидата
9. как ведётся учёт времени: JIRA или что-то ещё заполнять

проект

1. технологии проекта: 3D графика, ИИ, большие данные, высокие нагрузки, pixel perfect, мобильный клиент, ГИС, state management, интернет/локальный, blockchain
2. как давно развивается проект в котором я буду участвовать
3. версии фреймворка, библиотек
4. как работаете над качеством кода
    1. как оцениваете технический долг, как часто его оцениваете, как к нему относится Заказчик

    2. есть ли правила улучшения кода - соглашения, линтеры
    3. принято ли исправлять ошибки(рефакторить, подготавливать код) перед тем как реализовывать новый функционал
5. на сколько спринтов вперёд готова аналитика/дизайн
6. когда наступает/заканчивается новая/текущая веха
7. есть ли у вас документация, и как часто её обновляете: тест сценарии, дизайн руководства, бизнес логика, описание API, соглашения по коду
8. есть ли в проекте согласованная архитектура, подходы к именованию и расположению файлов
9. что буду делать из перечисленного: рефакторинг, проектирование, документирование, программирование, тестирование, вёрстка, дизайн, devops(webpack/gulp/docker), администрирование(linux/nginx)
10. какая доля кода написана людьми уже ушедшими из проекта, как давно они ушли

инструменты

1. Время сборки проекта
2. Какие технологии используются для доступа к стендам: VPN, RDP, WEB, AZURE, AMAZON
3. Какие ОС используют разработчики: Mac, Win, Linux
4. какие инструменты используют разработчики: VSCode, JIRA, SWAGGER, CONFLUENCE, FIGMA
5. какие инструменты используете для общения: skype, hangouts, slack, teams
6. Какие среды развёрнуты для разработчиков: test/QA/stage/dev
7. какие средства автоматизации дизайна/сборки/тестирования(UI, API): postman, selenium, jasmine, vagrant, chef, jenkins
8. какие средства совместной работы над аналитикой/дизайном: figma, draw.io, google docs, confluence

методики

1.
2. как договариваетесь про дизайн UI и контракты API: кто делает, согласует, в каком виде и где фиксируются, примеры и описание
3. как оцениваете вклад разработчиков в проект, их скорость/качество работы
4. как и кто будет меня оценивать, как он поймёт, что я хорошо или плохо прошёл испытательный срок
5. что используете из гибких методологий работы: спринты, ретроспективы, вехи, стендапы, доски, покер
6. что входит в описание задачи: макет, API, типы данных, сценарии, схемы
7. как принимаете решения по добавлению задач в спринт, как часто меняете/сдвигаете задачи после начала спринта
8. есть ли правила ранжирования важности задач и ошибок
9. есть ли правила проверки кода(code review) и работы с репозиторием(commit, merge), какие используются подходы git-flow, github-flow.

коллеги

1. есть ли у вас отдельные аналитики(BA)/тестировщики(QA)/админы(devops)
2. кто может регулярно и охотно отвечать на мои вопросы по коду и предметной области
3. кто в команде разработки: роль, опыт текущей роли(лет)

4. кто в команде работает удалённо(не в офисе)

Признаки здорового проекта

1. на проекте можно работать удалённо под linux/mac(есть впн, docker, облако)
2. в проекте нет текучки кадров
3. руководство проекта понимает необходимость сдвижки сроков выполнения задач, есть понятные и чёткие правила на этот счёт
4. Для оценки команды используется не только попадание в сроки, но и пропускная способность(кол-во задач в неделю), и качество кода(% без ошибок)
5. по метрикам в JIRA/ESLint видно, что возможно поддерживать разумные сроки выполнения задач
6. самые опытные разработчики довольны кодом, который они пишут и поддерживают
7. нет регулярной необходимости в сверхурочной работе, а если она возникает, то нет проблем с оплатой по сверхурочному тарифу или компенсацией выходными
8. руководство проекта умеет прояснять и обсуждать/корректировать свои ожидания, проводит совещания вежливо и конструктивно
9. вход и выход задач контролируется руководством проекта не поштучно, а за период. Задачи поштучно контролируются, при необходимости, техлидом
10. есть управляемый процесс удаления устаревшего/неиспользуемого функционала
11. есть процесс оценки качества кода проекта в целом
12. существующих инструментов(сервисов, виртуальных серверов) для командной работы достаточно, работают бесперебойно
13. руководство проекта прислушивается к просьбам/проблемам разработчиков, реагирует конструктивно и без волокиты