PHP

Серверная часть: 1) яп, 2) бд.

Требования к проекту:

* Клиентская часть на react ts.
* Адаптивная верстка (мультимедийность).
* Расположен в репозитории.
* Уникальный проект на группу (придумать коллегиально (допустимо разделение по ролям между группы)).
* Групповой проект ведется в одном репозитории, в котором ведется совместная разработка.

Допустимые роли в проекте:

* Бекендер
* Фронтедер
* Дизайнер
* Аналитик
* Тимлид
* Тестировщик

Аналитик, дизайнер и тимлид по обстоятельствам должны быть в единственном экземпляре.

Любой проект начинается с документа концепции, в котором выделяются основные моменты проекта, такие как:

* Цель и суть проекта
* Описание основного функционала
* Список исполняемых платформ (минимум 2)
* Основная фишка приложения.
* Краткое описание лора.
* Дальнейшее развитие.

Следующий документ требования или техническое задание, в котором детально описывается требуемый функционал.

Инструменты и организация работы .

Деление периода разработок на спринты, в которых задаются, разделяются и выполняются задачи.

Все разработки ведутся в миро.

Таск-менеджер:

* Состав команды с описанием ролей и ссылкой на гитхаб.
* Ссылка на репозиторий, в котором велась работа.
* Ссылка на разрабатываемые дизайны. Дизайнер разрабатывается дизайны в Figma.

Колонки доски ведения разработки (слева направо):

* Надо делать
* в работе
* сделано / тестируется
* готово к релизу
* в релизе

карточки в таск-менеджере:

* уникальный номер
* название
* описание
* исполнитель

тимлид явно разрешает pull-request в ветку готового проекта.

Каждая задача выполняется в репозитории в отдельной ветке, которая клонируется от ветки develop.

В ветке develop в каждый момент времени лежит рабочая версия приложения. В ветку develop имеет право пушить только тимлид, а точнее разрешает pull-request.

Каждая задача делается в новой ветке.

Новая ветка в проекте должна содержать номер задачи.

Пример названия ветки разработки задачи: project-123-make\_sth

Название\_проекта - номер\_задачи – название задачи

После того как задача сделана, разработчик спуливает все последние изменения dev ветки и разрешает конфликты.

После этого разработчик переносит задачу в готово-к-тестированию,

Тестировщик переключается в эту ветку, находит баги, оформляет баги соответствующими задачами в таск-менеджере. Когда все баги успешно исправлены, разработчик переносит задачу в готово-к-релизу, и оформляет pull-request для тимлида, после этого тимлид осуществляет пул-реквест или нет.

Единственный эффективный способ организации проекта для того чтобы проект сделался и работал.

Тимлид отвечает за соблюдения процесса разработки.

Работа с репозиторием.

Инструкция по работе:

* переключаетесь на ветку разработки (git checkout develop)
* спуливаете все изменения из develop (git pull origin develop)
* создаем свою ветку (git checkout -b BDSM-123-pain)
* коммит всех изменений (

git add .

git commit -m “…”)

* опять запулить в дев ветки (git pull origin develop)
* пушим в репозиторий нашей ветки разработки (git push origin BDSM-123-pain)
* тимлид делает или не делает пулреквест в гитхаб

Концепция требования, описания апи, описание дизайнов, описание структуры бд, должны лежать в репозитории в файле с расширением \*\*\*.md.

Дезешечка: написать состав комманд с распределением ролей, скинуть ссылку на репозиторий и быструю версию документации.

Скачать прогу open server.