Требования

Клиентская часть на react + typescript

Проект должен быть расположен в репозитории (один репозиторий и на клиент и на сервер)

Каждая группа реализовывает один проект в рамках группы(один проект на всех)

Тематика проекта любая

В рамках работы над проектом допустимо разделение команды по ролям

Групповой проект ведется в одном едином репозитории

Допустимые роли на проект:

Бекенд

Фронтенд (обязательно адаптивная вёрстка)

Дизайнер1ч

Аналитик 1ч

Тимлид 1ч отвечает полностью за проект

Тестировщик

Любой проект начинается с концепции (цель и суть проекта, описание основного функционала, список платформ на которых он исполняется, основная фишка приложения, краткое описание лора)

На основании концепции аналитик пишет требования, либо тз

Описывается детально требуемый функционал, так чтобы было понятно и заказчику, и исполнителю

Инструменты и организация работы

Вся разработка и все задачи должны вестись в программе Mirror

Там должно быть:

Название приложения

Состав команды и записанными ролями и с именем учетки в гитхабе

Должна быть прописана ссылка на репозиторий с проектом, ссылка на дизайны в фигме

Доска разделённая на колонки:

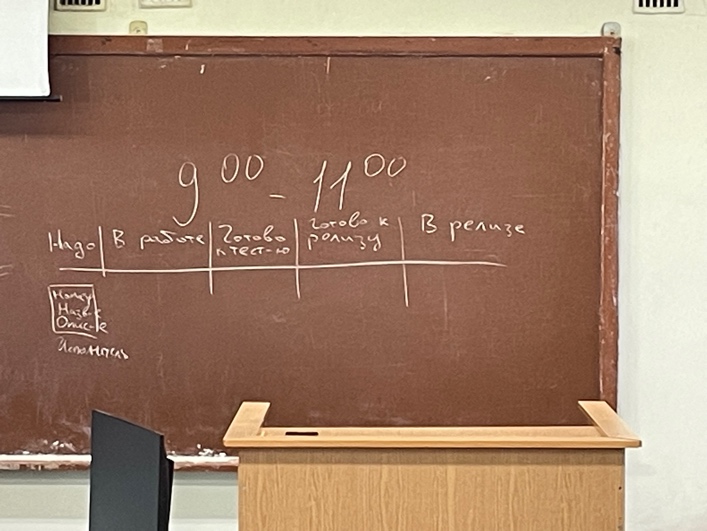
Надо делать

В работе

Можно тестировать

Готово к релизу

В релизе



Алгоритм работы – единственный способ организации работы в команде

Каждая задача выполняется в репозитории в отдельной ветке которая клонироутеся от ветки dev (основное название проекта-номер-название ветки), после того как задача сделана разработчик спуливает все последние изменения из dev и разрешает конфликты которые могут быть, после этого разработчик перетаскивает задачу к готово к тестированию, тестировщик переключается в эту ветку тестирует задачу, находит баги, оформляет баги следующими задачами, когда все баги успешно исправлены тестировщик или разработчик перетаскивает к готово к релизу и оформляет пул реквест на тимлида

В проекте две основные ветки main и dev

В ветке dev в каждый момент времени лежит рабочая версия приложения, в ветку dev пушить может только тимлид (не пушит, а разрешает pull-request)

Работа с репозиторием

Git checkout -b develop создать ветку

git checkout develop переключиться на ветку

git pull origin develop. Спулить изменения с девелоп

git checkout -b BDSM-123-pain создать свою ветку

git add .

git commit -m «комментарий».

Git pull origin develop

Решить конфликты

Git push origin BDSM-123-pain

Аналитик

Концепция требования api описание дизайна и тп должны лежать в этом же репозитории в файлах md

К следующей лекции написать сосавы команд и списком ролей, ссылку на доску и репозиторий, концепция

Поставить программу open server

16.09 Синтаксис PHP

$\_1

$1a

PHP очень требователен к наличию точек с запятой

В PHP нету типа данных undefind, null есть

Конкатенация строк в php осуществляется с помощью инструкции точка.

$a=’Вася’;

$b=’Маша’;;

$a.$b=’ВасяМаша’;

В PHP из коробки в строку мы можем вставить значение переменной.

$arr=[ ]; - массив

$arr[]=123; - добавить элемент в массив

Ассоциативные массивы != объектам

$arr=array(

‘name’=>9,

‘age’=>18);

$arr=[‘name’]

Объекты в PHP

$obj = new stdClass();

$obj->key1=’Вася’;

Классы

class Point{…}

Наследование с помощью команды extends

Class Name {

Public

Protected

Private}

Модификатор доступа может быть задан для любого Элемента. По умолчанию используется значение public.

Public – позволяет обратиться к методу или свойству класса в экземпляре класса (снаружи) и в классе потомке к методу или свойству родителя.

Protected – запрещает вызов метода в экземпляре класса,

Private – protected, только дополнительно запрещены вызовы методов классов родителей, в классе потомка.

У классов  PHP есть метод конструктора и деструктора.

Function\_\_construct(){

}

Function decstruct(){

}

Parent::\_\_construct();

В PHP можно управлять вызовом конструктора родителя, потому что вызов конструктора родителя является необязательным.

Конструктор родителя можно явно вызвать в конструкторе потомка, в случае если его не вызвали, он не будет вызван. Если в родителе конструктор есть, а в потомке конструктора нет, то в потомке конструктор родителя будет вызван автоматически.

Статичный класс – класс методы которого могут быть вызваны без создания экземпляра класса.

$this->

Супер глобальная переменна - $\_GET

$\_POST параметры постового запроса

У гетового запроса отсутствует тело запроса, и его параметры передаются в адресной строке. У постового запроса есть body, и его параметры мы можем передать не только в адресной строке, но и в body.

Для того чтобы написать бекенд мне не нужен клиент.

В запросах могут быть заголовки, к запросу могут быть прикреплены куки файлы.

Основной способ аунтефикации пользователя, это его определение по специализированному ключу – токену, который имеет ограниченный срок жизни и выдается клиенту на 1 сессию.

Для снижения нагрузки на бекенд применяется механизм кэширования, суть которого заключается в следующем: каждый новый запрос в бекенд, обрабатывается, на него формируется ответ и этот ответ складывается в так называемый кэш и в случае прихода в бекенд аналогичного запроса бекнд отвечает на него ответом из кэша.

Принцип работы бекенда на проекте:

Любой запрос на бекенд должен быть обработан единообразным способом, и на него должен быть дан однообразный ответ.

Result:’ok’|’error’,

Data?:any

Error?:{

Code:number,

Text:string

}

В проекте должен быть api, где описаны все запросы которые может обработать бекенд(форматы ответов и возможные ошибки)

Порядок работы:

Если необходимо написать новый метод на бекенде, сначала он описывается в файлике api.md, описывается адрес метода, его параметры, ответы, возможные ошибки.

ДЗ: на основании проекта необходимо разработать документацию по методам необходимым в рамках проекта.

На клиенте необходимо разработать структуру страниц клиента и расписать базовый функционал.

***30.09 Разбор NoPaneNoGame***

Репозиторий состоит из двух папок: client и server.

В файле readme написана информация о развертывании проекта и его запуску.

Файл API – документ описывающий методы бекенда.

При разработке нового сервисного метода он сначала описывается в файле api, и только после этого пишется код. Ответственность за ведение документа api лежит на бекенд разработчике. Тимлид имеет право не принимать pull-request на новый метод, при отсутствии его описания в документе.

Бекенд.

Для запуска бекенда необходимо всё содержимое папки server скопировать в соответствующую папку в openServer. Либо можно настроить openserver, чтобы он целился в папку на проекте. Бекенд всегда возвращает ответы в виде json файлов. Бекенд спроектирован в парадигме MVC (Model – бизнес логика приложения, View – представление, Controller – связующее звено). Модели не знают о представлениях, а представления не знают о модели. Представления должны быть тупыми, не содержать никакой бизнес-логики. Модели не знают как они представляются и работают только с данными.

Структура бекенда.

Весь бэкенд

Api/

Index.php

Application/

Модули

Answer.php

Application.php

Файл index.php является единственной точкой входа в php, в нём мы определяем формат возвращаемого ответа и список поддерживаемых методов, также там разрешены кросс-доменные запросы.

В index.php есть два header один опрелделяет формат ответа второй разрешает кросс-доменнные запросы

Подключение файлов в php осуществляется множеством способов, в частности будем использовать метод requireOnce.

Функция result определяет какие методы поддерживает бекенд и каким методом из application этот входящий запрос будет обработан.

Каждый новый метод сначала объявляется в index.php. Index.php формирует ответ с помощью команды echo, функции json\_encode и метода response класса answer.

Answer.php

В бэкенде все реализовано на клссах, кроме answer.php.

В answer.php объявлен класс отвечающий за формирование ответа, также в нём расписаны все возможные коды ошибок.

Коды ошибок задаются в виде ассоциативного массива, где ключ это код ошибки (число), а значение это текст ошибки.

Статический метод response в answer проверяет следующее:

1. Если переданные в него данные не false и это данные которые содержат ассоциативный массив с ключом error, в противном случае возвращается успешный ответ
2. Если данные никакие не переданы, то ошибка 9000

Статические методы не имеют доступа к контексту класса.

В php вызов cтатических методов или обращение к статическому свойству осуществляется через ‘::’

По концепции MVC класс answer является представлением, и ещё частично файл index.php.

В index.php в функцию result передаётся аргументом супер-глобальная переменная get.

Application.php

Класс, выполняющий функцию контроллера. В конструкторе этого класса объявляются экземпляры всех моделей: user и chat.

Также в коснтрукторе класса создается экземпляр для работы с бд и этот экземмпляр передаётся в каждую модель. Передавать модели друг в друга запрещено!

В Application для каждого запроса создаётся свой метод. В application каждый метод как обработчик запроса проверяет и валидирует входящие параметры запроса и в случае если они хорошие вызывает соответствующий метод соответствующей модели. Если модели для работы необходимы данные для работы, то эти данные получаем в application.

Метод sendMessage.

Этот метод отвечает за отправку сообщения в чат. Сначала проверяется наличие параметров запроса – параметров token и message. В случае если их нету, возвращается ошибка. Потом из модуля user мы получаем пользователя по токену.

В случае если такого пользователя нет, возвращается другая ошибка. Если пользователь есть, то вызывается метод чата sendMessage. Который передаёт идентификатор пользователя и его сообщение. В модели chat, в методе sendMessage сначала добавляется в базу данных новое сообщение, вторым методом обновляется hash сообщений.

Авторизация осуществляется с помощью метода login. В этом метод мы передаем Login, hash – может быть использована любая hash-функция, в частности md5(md5(логин+пароль)+rnd), rnd – случайное целое число.

В базе данных в поле пароль хранится хэш от логина+пароля. Пароль в бд в явном виде хранить – катастрофический мовитон.

Процесс авторизации.

По логину из бд мы получаем пароль добавляя к нему параметр rnd мы берем от него hash и получаем контрольную сумму. Которая сравнивается с переданным хешем с клиента. Если они равны то создаётся токен для этого пользователя, он записывается в базу данных и вместе с информацией по пользователю возвращается клиенту.

Logout

По токену берем пользователя из бд, если он нашелся – токен очищаем.

Сохранённые данные между запросами не сохраняются. Единственный способ сохранить данные между запросами – записать их в бд.

ДЗ: на бекенде сделать новую модель ‘PainOfDomashka’ туда перенести методы derivative, касатльная к производной,

Клиент должен поддерживать логин логаут и построений графиков клиент на реакте.