## Общие положения

1. Задачи необходимо выполнить в системе контроля версий git. Можно использовать любую систему по собственному усмотрению, например GitLab. Необходимо создать собственный публичный проект и отправить ссылку на него после выполнения задач на email [hello@trimium.ru](mailto:hello@trimium.ru).
2. Проект необходимо начать с добавления этого .docx файла в репозиторий.
3. **Выполнение всех заданий, описанных в этой тест-анкете не требуется**, достаточно выбрать по 1-2 из каждого раздела. На остальные задания необходимо ответить, чтобы мы могли понимать уровень ваших навыков, для этого отредактируйте данный файл выделив цветом один из вариантов ответа по каждой задаче, по необходимости снабдите пояснением (например, как вы это видете, как лучше сделать). Отправьте коммит в репозиторий.
4. Задачи, выбранные для выполнения, выполняются в этом же репозитории.
5. Инструкции по запуску, структуре файлов, каталогов, список необходимого ПО для оценки выполнения заданий описать в README.md в корне проекта.

## Общие требования

1. Код должен быть читаем и без комментариев. Без русских слов в транскрипции. Без сокращений за исключением общеизвестных аббревиатур. Только английские слова, отражающие суть;
2. Не человекочитаемые фрагменты кода (запросы, циклы, регулярные выражения) обязательно сопровождаются комментарием;
3. Работать в IDE. Все объявленные классы, свойства, функции должны выбираться из списка, предлагаемым ide. Для функций описывать типы входных параметров и результата. Для каждого использования класса, свойства, переменной должна быть возможность перехода к объявлению, другому использованию в коде (например по ctrl+lmb), доступна быстрая справка и тип данных;
4. Использовать только именованные константы, перечисления (if (state == 1) недопустимо);
5. Стиль кода в соответствии со стандартами используемого фреймворка;

## Бэкенд (любой язык php, node, python)

1. Установить и запустить фреймворк (как угодно. php-fpm, nginx, cil sapi, apache2… В README.md описать и оставить только инструкцию для запуска проекта. Использовать субд mysql (innodb);  
   Написать импорт из прилагаемого [csv](https://drive.google.com/file/d/1U1EVFDvWQDuD3FsGiPSvJAp9K6x9Mx9B/view?usp=sharing) файла (пользователи и их профили). Необходимо реализовать минимальный скрипт чтения из файла и записи в бд. Импорт должен запускаться из cli и выводить информацию о ходе процесса;
   * выполнить (обязательно к выполнению)
2. Реализовать возможность запустить импорт с помощью загрузки файла через форму в браузере по http;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
3. Реализовать REST api для получения списка профилей в формате json, с возможностью изменить сортировку и дозагрузки (постранично);
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
4. Реализовать GraphQL api аналогично предыдущему пункту.
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
5. В каждый профиль необходимо добавить свойство с количеством профилей, проживающих в его городе. Необходимо обеспечить скорость выполнения запроса используя систему memcached.
   * Выполню (используется стандартное кэширование, к которому можно прикрутить memcached в конфигах)
   * знаю, могу
   * могу освоить
   * не могу выполнить

## Бэкенд (mysql)

Пользователь портала может заполнить профиль, запостить видео, изображение, документ, ссылку на веб-ресурс, свое местоположение, лайкать посты других пользователей, оставлять комментарии постам, чужим комментариям.

Необходимо описать структуру (DDL) бд для вышеописанной ситуации.

1. Добавить ERD (можно скрин) в репозиторий.
   * выполню (отправлю коммит в git)
   * знаю, могу (создать таблицы: user, post, post\_image, post\_document, post\_link, post\_coordinate, post\_like, post\_comment)
   * могу освоить
   * не могу выполнить
2. Реализовать наследование сущностей. Посты могут быть разных типов (видео, изображение, документ, ссылка на веб-ресурс, геопозиция). Каждый тип обладает как уникальным набором полей (ссылка на видео, ресурс, файл, координаты), так и общими (тип поста, автор, время). Необходимо написать inheritance.sql содержащий ddl (структуру таблиц, связи);
   * выполню (отправлю коммит в git)
   * знаю, могу (Таблицы, имеющие префикс post, привязываются к таблице post один-ко-многим. Таблица post имеет общие для всех ресурсов поля: автор, время)
   * могу освоить
   * не могу выполнить
3. Реализовать древовидную структуру. Комментарии постов могут иметь иерархию. Необходимо реализовать подсчет кол-ва всех дочерних комментариев и обеспечить выборку поддерева относительно комментария по id наименьшим количеством запросов (отдельно кол-во, отдельно строки). Необходимо написать tree.sql содержащий структуру таблиц, связи и выборку относительно id одного комментария;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить (Комментарии должны иметь поле parent\_id. У корневых комментариев он равен нулю, у дочерних – имеет id родительского комментария. SQL-запрос по подсчёту комментов с неограниченным деревом не напишу. Возможно, в гуле есть)
   * не могу выполнить
4. Реализовать выборку самых популярных (по кол-ву лайков постов) авторов на каждый месяц за последний год для отображения в графике (динамика популярности). Необходимо написать profiles.sql содержащий структуру таблиц (профиль, пост, лайк), связи и выборку за последние полгода;
   * выполню (отправлю коммит в git)
   * знаю, могу (Не проверял:

select u.\*, sum(l.likes) as total\_likes

from post p

join (

select post\_id, user\_id, count(\*) as likes

from post\_like

group by post\_id

) l on l.post\_id = p.id

join user u on u.id = p.user\_id

group by l.post\_id

where p.date\_create between :from\_date and :to\_date

order by total\_likes desc

* + могу освоить
  + не могу выполнить

## Бэкенд (администрирование)

1. Написать bash скрипт для установки необходимого для работы вебсервера (ubuntu LTS) по: php-fpm, php-cli, php-mysql, nodejs + npm, python, mysql (mariadb), nginx;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить (могу только в стиле sudo apt-get install <program>, но там ещё надо конфиги крутить, чтобы это быстрее работало. Гемор, в общем, лучше использовать настроенные docker-контейнеры, но их нужно долго настраивать.
2. Организовать доставку статических файлов фронтенда с кешированием и проксирование api бэкенда без кеша, доступным на одном доменном имени и порту, с помощью nginx;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить (проксировать запросы api бэкенда можно и через angular-cli)
   * не могу выполнить
3. Сгенерировать самоподписанный сертификат. Настроить редирект с http на https с использованием сертификата в конфиге nginx;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить (OpenSSL для генерации сертификатов (команды есть в шпаргалке), редирект на https – методом гугления. Такие операции делались раз в жизни и в голове не откладываются)
   * не могу выполнить
4. В корне проекта описать Dockerfile, содержащий инструкции для установки и настройки в контейнере ПО и конфигов из задач выше за исключением СУБД;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить (но не самостоятельно, а на основе чужих проектов)
   * не могу выполнить
5. Описать docker-compose.yml для продакшн и отдельно для разработки с разным окружением (переменными) и отдельным контейнером для СУБД.
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить (но не самостоятельно, а на основе чужих проектов)
   * не могу выполнить

## Фронтенд

* + Каждое задание не подразумевает визуального оформления (css) компонентов. Задачи нацелены на получения сведений о навыках программирования.

1. Использование для выполнения задач популярного фреймворка/библиотек (angular, vue, react…);
   * выполню (Angular 2+)
   * знаю, могу
   * могу освоить
   * не могу выполнить
2. Использовать TypeScript;
   * выполню
   * знаю, могу
   * могу освоить
   * не могу выполнить
3. На странице /profile/1 необходимо отобразить страницу с шапкой, содержащей ФИО и изображение профиля текущего пользователя, и формой для редактирования профиля с id указанным в url. Форма имеет поля: изображение (файл), фио (3 текстовых поля), руководитель - ссылка с фио и изображением. При выборе файла сразу меняется изображение профиля на выбранное. При клике по ссылке открывается диалоговое окно с такой же формой, но редактированием руководителя (может быть несколько окон). При изменении изображения или фио оно автоматически обновляется в форме подчиненного и в шапке если был отредактирован профиль текущего пользователя.
   * Выполню (частично выполнил, кроме отображения новых окон)
   * знаю, могу
   * могу освоить
   * не могу выполнить
4. Сохранение профиля (отправка данных на сервер) происходит автоматически при внесении изменений. Необходимо отправлять запросы не чаще одного раза в 3 сек. Для реализации использовать библиотеку rxjs
   * выполню
   * знаю, могу
   * могу освоить
   * не могу выполнить
5. При открытии ссылки руководителя в отдельной вкладке (клик средней кнопкой мыши/контекстное меню) на странице отображается форма профиля руководителя. При изменении профиля в этой вкладке ссылка на руководителя в родительской вкладке должна обновиться
   * выполню
   * знаю, могу
   * могу освоить (Не совсем понял, что нужно сделать. Что имелось в виду под родительской вкладкой: речь идёт о модальных окнах в пределах одной вкладки или это предыдущая вкладка, которая открыла текущую)
   * не могу выполнить
6. В форме для текущего пользователя добавить кнопку “сделать фото” при клике на которую пользователь сможет получить изображение с веб-камеры.
   * выполню
   * знаю, могу
   * могу освоить
   * не могу выполнить