## 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.

## Фронтенд (angular)

Каждое задание не подразумевает визуального оформления (css) компонентов. Задачи нацелены на получения сведений о навыках программирования и знания фреймворка.

1. Установить и запустить Angular 8+ с помощью npm;
   * выполню (обязательно к выполнению)
   * знаю, могу
   * могу освоить
   * не могу выполнить
2. Создать простой макет (компонент layout) с шапкой, навигационным меню и местом для контента. Настроить навигацию с использованием макета и компонентом PageNotFound в качестве контента;
   * выполню (обязательно к выполнению)
   * знаю, могу
   * могу освоить
   * не могу выполнить
3. Сгенерировать модуль “social-network” с навигацией и динамической загрузкой;
4. По ссылке social должен быть доступен список **постов** всех пользователей (массив объектов, описанный вручную в отдельном файле);
   * выполню (обязательно к выполнению)
   * знаю, могу
   * могу освоить
   * не могу выполнить
5. Написать сервис, загружающий список **постов** по rest api (ajax);
   * выполню (обязательно к выполнению)
   * знаю, могу
   * могу освоить
   * не могу выполнить
6. По ссылке /post/:id необходимо отобразить компонент отображающий фио автора и содержимое поста в зависимости от его типа (видео, изображение, документ, ссылка на веб-ресурс, свое местоположение);
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
7. Пост должен загружаться до рендеринга компонента (pre-fetching. Параметр resolve в конфиге Route). Аналогично списку постов брать из файла/rest/gql;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
8. Сортировка двух “бесконечных” списков. Имеются 2 списка профилей пользователей. Слева все пользователи, справа заблокированные. Нужно реализовать оба списка с виртуальным скроллом и возможностью перетащить профили из одного списка в другой. Для реализации использовать Angular Material Component Dev Kit (CDK)
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
9. Динамические компоненты. Имеется объект {html: ‘<h1>Привет <user></user></h1>’, user: {lastName: ‘Иванов’, firstName: ‘Иван’, pictureUrl: ‘/picture?id=69’}}. Необходимо отобразить html из объекта на странице, а на месте <user> должен быть рабочий компонент UserComponent, отображающий аватарку и фио пользователя.
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить

## Верстка

1. Подготовить окружение для разработки проекта:
   1. автоматическое обновление контента в браузере при изменении исходников;
   2. установить sass (scss)/less компилятор;
   3. подключить используемые библиотеки (bootstrap, tailwindcss…);
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
2. Разработать файл конфиг, содержащий объявления переменных и их значения по-умолчанию. Подразумевается, что этот конфиг будет генерироваться программой в соответствии бренду заказчика;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
3. Подготовить альтернативный конфиг с другим цветовым решением;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
4. Сверстать лейаут: шапка с логотипом и пользователем, навигационное меню;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
5. Сверстать список постов. Посты могут быть разных типов (видео, изображение, документ, ссылка на веб-ресурс, свое местоположение).
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить

## Бэкенд (node)

1. Установить и запустить фреймворк express. В README.md описать и оставить только инструкцию для запуска проекта. Использовать субд mysql (innodb);
   * выполнить
   * знаю, могу
   * могу освоить
   * не могу выполнить
2. 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;
   * выполню (отправлю коммит в git)
   * знаю, могу
   * могу освоить
   * не могу выполнить
3. 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
4. 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
5. Реализовать GraphQL api аналогично предыдущему пункту. Выбрать реализацию самостоятельно (REST/GQL).
   * выполню (отправлю коммит в git)
   * 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
   * 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
   * 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
   * 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
   * могу освоить
   * не могу выполнить