**Hello ~~world~~ наш потенциальный сотрудник!**

Перед вами тестовое задание по подбору в команду Manao.

*На первый взгляд оно может показаться большим и сложным, но это только лишь из-за того, что мы постарались максимально подробно расписать задание,*

*оставили подсказки, ведь мы заинтересованы в его выполнении не меньше вашего! =)*

*Удачи! Ждём на собеседовании! ;)*

**Создание веб-приложения** для регистрации и авторизации на php (не framework), html, css, js/jquery.

\* можно использовать пакетный менеджер (composer) и php шаблонизаторы (twig, blade), но без самих фреймворков.

**Форма регистрации**

1. Поля:
   * login (unique)    [валидация : минимум 6 символов ]
   * password          [валидация : минимум 6 символов , обязательно должны состоять из цифр и букв]
   * confirm\_password
   * email (unique)    [валидация : email]
   * name              [валидация : 2 символа , только буквы]
2. Функционал:
   * после заполнения формы, уходит запрос на создание пользователя;
   * поля проверяются на валидность, уникальные поля проверяются на уникальность в БД, password и confirm\_password сравниваются (должны быть одинаковыми);
   * если что-то не так - выдается ошибка. Возвращаем текст с ошибками. Подставляем их рядом с полями, в которых допущена ошибка.
   * в случае успеха шифруем пароль ("соль"+md5 или "соль"+sha1) и заносим данные в нашу базу.

**Форма авторизации**

1. Поля:
   * login
   * password
2. Функционал:

После заполнения формы уходит запрос:

* проверяется наличие пользователя в бд;
* сверяется пароль.

    В случае ошибки, возвращаем текст с ошибками. Подставляем их рядом с полями, в которых допущена ошибка.

    В случае успеха происходит процесс авторизации (создание необходимых связей cookies + session ).

    В случае если пользователь авторизован, на всём сайте мы должны видеть сообщение 'Hello [имя пользователя]' и кнопку "Выйти", формы в таком случае не отображаем.

    Внешний вид значения не имеет.

**Основные требования**

* Приложение обязательно должно быть написано используя ООП концепцию.  
   **\*** Должны быть выделены отдельные сущности, функционалы и для них описаны классы.
* Вместо базы данных используем файл JSON. Другие БД/технологии не интересны.

\* <https://ru.wikipedia.org/wiki/JSON>

\* <https://www.php.net/manual/ru/ref.json.php>

* Файл базы не должен быть доступен по прямому урл  
  \*  [при использовании веб-сервера apache](https://ru.wikipedia.org/wiki/.htaccess)\*  [при использовании веб-сервера nginx](https://nginx.org/ru/docs/http/ngx_http_core_module.html#location)
* Доступ к файлу базы должен осуществлятся только на стороне php.  
  \* имеется ввиду, что при выполнении предыдущего пункта, к файлу нельзя будет обращаться, например, через ajax запрос.
* Для работы с БД должен быть описан класс реализующий CRUD
* \* [CRUD](https://ru.wikipedia.org/wiki/CRUD) - это акроним, описывающий функционал Create / Read / Update / Delete. Т.е. в этом пункте, мы просим создать инструмент на ООП для работы с нашей базой.
* Пароль не должен хранится в открытом виде.
* Связь авторизации  - cookies + session

\* когда происходит авторизация, мы должны запомнить нашего пользователя, чтобы при переходе на следующие страницы, мы смогли его идентифицировать и показать, что он авторизован.

* Формы должны отправляться используя технологию [ajax](https://ru.wikipedia.org/wiki/AJAX), ответ в формате json.

\* <https://learn.javascript.ru/fetch>

* Файлы обработки ajax-запросов не должны быть доступны из браузера, если это не ajax-запрос.

 \* Простая проверка, если к нам пришёл не ajax-запрос, то работу не выполняем.

* Если js отключен, то форма отправиться не может.
* Все поля форм - обязательные.
* Все формы реализуются через <form>.
* Результаты залить на github/bitbucket. Сделать репозиторий публичным. Ссылку прислать по почте.

**Дополнительные требования**

    \*\* Эти требования не обязательны для исполнения, но будут большим плюсом при выполнении.

* защита от инъекций при получении параметров через get- или post;
* осмысленное название полей в бд;
* использование PSR;
* соблюдение принципов SOLID;
* разумное оформление кода – имена переменных, методов, классов, отступы, комментарии;
* код должен быть прост в поддержке;
* отсутствие дублирование кода. Копипаст – признак низкой культуры разработки;
* разделение логики работы и представления – некрасиво писать html и php вперемешку;
* css/js в отдельных файлах;
* осмысленное использование css.

Данное задание будет проверятся на сервере linux с php 7.4 , разрабатывая на windows не забывайте про написание слэшей в правильную сторону! =)

И различия в использовании верхнего и нижнего регистра наименования файлов.

**Какие тесты должно пройти задание:**

**Общие проверки**

1. Проверить обязательность полей.
2. Проверить реализацию форм.
3. Проверить приложение на соответствие концепции ООП.
4. Проверить использование файла JSON вместо БД.
5. Проверить отсутствие доступа к файлу БД по прямому URL.
6. Проверить хранение пароля в зашифрованном виде.
7. Проверить переход по страницам сайта без повторной авторизации (cookies+session).
8. Проверить отправку формы с использованием AJAX.
9. Проверить соответствие ответа формату JSON .
10. Проверить невозможность отправки формы с отключенным JS.

**Форма регистрации**Общие проверки:

1. Проверить обязательность полей.
2. Проверить наличие текста ошибки при вводе невалидных данных.

Поле login:

1. Оставить поле незаполненным и попытаться отправить форму.
2. Проверить валидность длины логина.
3. Проверить ввод логина с пробелами между/в начале /в конце символов.
4. Проверить отправку формы только из пробелов.
5. Проверить отправку формы  уже зарегистрированного логина в БД.

Поле password:

1. Оставить поле незаполненным и попытаться отправить форму.
2. Проверить шифрование пароля при вводе (\*\*\*\*\*\*\*\*).
3. Проверить валидность длины пароля.
4. Проверить ввод валидного пароля.
5. Проверить ввод пароля с пробелом между/ в начале / в конце / символов.
6. Проверить ввод валидного пароля + спец.символы.
7. Проверить отправку формы только из пробелов в поле.

Поле confirm\_password:

1. Проверить ввод пароля отличного от пароля в поле password.
2. Проверить ввод пароля совпадающего с паролем в поле password.

Поле email:

1. Оставить поле незаполненным и попытаться отправить форму.
2. Проверить ввод адреса с двойным символом @.
3. Проверить ввод адреса с пробелами между/в начале /в конце символов.
4. Проверить ввод адреса без доменной части.
5. Проверить ввод адреса без точки в доменной части.
6. Проверить ввод адреса уже зарегистрированного в БД.

Поле name:

1. Оставить поле незаполненным и попытаться отправить форму.
2. Проверить валидность длины имени.
3. Проверить границы длины имени.
4. Проверить ввод имени с пробелами между/в начале /в конце букв.
5. Проверить ввод имени только из пробелов .

**Форма авторизации**

Общие проверки:

1. Проверить шифрование пароля при заполнении поля password (\*\*\*\*\*\*\*).
2. Проверить наличие текста ошибки при вводе невалидных данных.
3. Проверить наличие сообщения “Hello [имя пользователя]” на каждой странице сайта при успешной авторизации.
4. Проверить создание cookies + session при успешной авторизации.

Поле login:

1. Проверить ввод валидного логина и невалидного password.
2. Проверить ввод валидного логина.
3. Проверить ввод логина незарегистрированного в БД.
4. Оставить поле пустым и отправить форму.

Поле password:

1. Проверить ввод валидного пароля для зарегистрированного пользователя.
2. Проверить ввод невалидного пароля для зарегистрированного пользователя.

**!!! Важно. Не добавляйте в readme.md условия данного тестового задания.**