**Инструкция к работе**

**Фронтенд**

Необходимо разработать приложение, состоящее из следующих основных блоков пользовательского интефейса:

*1. Основная страница приложения.*

Основная страница должна содержать:

* общую информацию для пользователя о приложении;
* кнопку или ссылку для перехода на форму регистрации пользователя с заданием минимальноо набора данных: логин, полное имя, e-mail, пароль.

При этом должны проверяться основные ограничения на значения этих полей:

*  логин только из латинских букв и цифр, первая буква, длина от 4 до 20 символов
*  e-mail должен соответствовать формату для адресов электронной почты (для проверки можно использовать регулярные выражения)
*  пароль должен содержать не менее 6 символов, включающий как минимум одну заглавную букву, одну цифру и один специальный символ.

При несоответствии требованиям должны отображаться соответствующие информативные сообщения в форме с возможностью исправления и повторной отправки на сервер.

* кнопку или ссылку для перехода на форму аутентификации с вводом и проверкой логина на наличие в БД и правильности пароля. Результат неуспешной проверки должен отображаться в форме с возможностью повторной отправки на сервер. При успешной аутентификации должен осуществляться переход на страницу в зависимости от прав пользователя в системе.

*2. Административный интерфейс системы для настройки её параметров и управления пользователями и их файловыми хранилищами.*

Вход в эту часть приложения доступен только пользователям, имеющим признак «администратор» в списке пользователей:

* список пользователей с выводом признака «администратор» и информации, введённой пользователем в форме регистрации (кроме пароля). В списке должна быть возможность удаления пользователей и изменения значения признака «администратор»;
* в списке пользователей должна также отображаться информация об их файловых хранилищах - количество и размер файлов, ссылка для перехода к интерфейсу управления файлами в хранилище.

*3. Интерфейс управления файловым хранилищем.*

Вход в интерфейс доступен для любых пользователей. При этом администратор должен иметь право управления хранилищами любых пользователей, включая своё собственное. Обычные пользователи должны иметь доступ только к своему хранилищу:

* в интерфейсе должен отображаться список файлов, загруженных пользователем в хранилище, с основной информацией о них: имя файла, комментарий, размер, дата загрузки, дата последнего скачивания;
* для каждого файла должны быть доступны операции: удаление файла, переименование файла, просмотр файла (средствами браузера или через загрузку на локальный диск), копирование специальной ссылки на файл для предоставления доступа другим пользователям или использования его в качестве ресурса в веб-приложениях;
* должна быть реализована возможность загрузить новый файл в хранилище с указанием комментария.

Общие требования к интерфейсу приложения:

* должна быть максимально использована концепция SPA (single page application), т.е. весь переменный контент на странице (списки пользователей и файлов и т.п.) должен формироваться кодом на JavaScript с использованием библиотеки React. Для получения данных должны использоваться асинхронные api-вызовы к серверу приложения;
* все страницы приложения должны содержать навигационное меню, формируемое в зависимости от состояния аутентификации пользователя (кнопки «Вход», «Выход» и «Регистрация»)
* не предполагается существенных усилий по оформлению приложения с использованием графики, внешних библиотек элементов и т.п.!!! Однако интерфейс приложения должен быть логичным и интуитивно понятным пользователю, имеющему опыт работы с аналогичными веб-приложениями

**Серверная часть приложения (бэкенд) должна соответствовать следующим требованиям:**

1. Реализация на Python с использованием фреймворка Django и использованием для хранения информации СУБД Postgres.
2. Настройки приложения, такие как параметры подключения к БД, размещения файлового хранилища и т.п., должны быть выделены в коде в отдельный модуль.
3. Загрузка статических ресурсов, таких как HTML, CSS, JS файлов фронтенда, а также api-вызовы, должны обрабатываться единым сервером.
4. В проекте должны быть созданы все миграции, необходимые для инициализации БД в работоспособное состояние - создание БД, таблиц, пользователя admin с правами администратора.
5. Все API-вызовы должны соответствовать семантическим правилам для REST API, для обмена данными между фронтендом и бэкендом используется формат JSON.
6. Сервер должен содержать реализацию бэкенда для двух основных блоков приложения: административный интерфейс, работа с файловым хранилищем.
7. Административный интерфейс должен содержать следующие функции (конкретные api-вызовы вам необходимо спроектировать самостоятельно):

*  регистрация пользователя (с валидацией входных данных на соответствие требованиям, описанным выше)
*  получение списка пользователей
*  удаление пользователя
*  аутентификация пользователя
*  выход пользователя из системы (logout).

Общие комментарии к этому блоку:

* данные о пользователях должны храниться в таблице(ах) БД в полях, имеющих соответствующие им типы: логин, полное имя, e-mail, пароль, признак администратора, путь к хранилищу пользователя относительно общего пути к хранилищу файлов
* все вызовы кроме регистрации пользователя должны быть защищены проверкой на то, что пользователь аутентифицирован в системе
* функция удаления пользователей должна быть доступна только пользователю, имеющему признак администратора
* ошибки должны возвращаться из api в виде соответствующих статус-кодов, а также в формате JSON.

1. Блок работы с файловым хранилищем должен содержать следующие функции (конкретные api-вызовы необходимо спроектировать):

* получения списка файлов пользователя
* загрузка файла в хранилище
* удаление файла из хранилища
* переименование файла
* изменение комментария к файлу
* скачивание файла
* формирование специальной ссылки на файл для использования внешними пользователями
* скачивание файла через специальную ссылку, используемую внешними пользователями или веб-приложениями.

Общие комментарии к этому блоку:

* все функции работы с хранилищем должны проверять права доступа пользователя к хранилищу
* для администратора должна быть доступна работа с хранилищем любого пользователя (функция получения списка файлов должна принимать параметр с указанием хранилища, если пользователь администратор).
* файлы должны сохраняться на диске сервера под уникальными именами в системе папок, не допускающей конфликты имен в случае, если разные пользователи загружают файлы с одинаковыми именами
* для каждого файла в БД должна сохраняться информация: оригинальное имя файла, размер, дата загрузки, последняя дата скачивания, комментарий, путь к файлу в хранилище, специальная ссылка, по которой файл может быть скачан внешним пользователем
* базовая папка для хранения файлов должна настраиваться как параметр системы
* специальная ссылка на файлы должна формироваться в максимально обезличенном виде, т.е. не содержать в себе имени пользователя, информации о хранилище пользователя, оригинальном имени файла
* при скачивании файла по такой ссылке он должен выгружаться сервером с указанием оригинального имени файла.

Общие требования к серверу:

* должно отслеживаться состояние аутентификации пользователя через сохранение информации о сессии
* все API-вызовы должны проверять права доступа пользователя и возвращать соответствующие ошибки через HTTP-статус и сообщение в формате JSON
* все события сервера должны логироваться путем вывода на консоль сообщений по типам DEBUG, INFO, WARNING, ERROR с указанием даты и времени, содержать информацию достаточную для анализа работоспособности и отладки сервера.