Используя NodeJS и СУБД PostgreSQL разработать приложение для вывода и редактирования списка произвольных сущностей.

Создать в БД сущность «Пользователь» с минимальным набором атрибутов:

* Идентификатор пользователя
* ФИО пользователя
* Логин пользователя
* Пароль

Создать в БД справочник «Роли доступа» с минимальным набором атрибутов:

* Идентификатор роли
* Название роли

Создать роли с именами ROLE\_LIST\_VIEW, ROLE\_ADD, ROLE\_EDIT, ROLE\_DELETE.

Создать в БД таблицу, содержащие список любых объектов с их характеристиками (выбор за Вами - список автомобилей в салоне, список книг в библиотеке, список продуктов в магазине, список сотрудников на предприятии и т.п.). Основные требования – таблица должна содержать как минимум по одному полю следующих типов:

* Текстовое поле длинной до 255 символов
* Текстовое поле длинной свыше 1000 символов
* Поле с целым целом
* Поле с дробным числом
* Поле с датой
* Поле с датой и временем
* Ссылка на простой справочник
* Ссылка на древовидный справочник
* Поле image (хранение графического файла)
* Поле blob (хранение произвольного файла)

Создать 2 справочника (простой [id, name] и древовидный [id, name, parent]), заполнить их любыми значениями по количеству не менее 10. Для древовидного справочника достаточно 2-х уровней вложенности.

Заполнить основную таблицу значениями по количеству не менее 50.

Для создания таблиц, вторичных ключей, индексов (если необходимо) и заполнения их данными использовать систему миграции (отразить в описании к тестовому заданию).

Создать простую страницу авторизации в системе по логину и паролю и разработать метод авторизации. В случае ошибки авторизации должно выводиться сообщение «Неверно указаны имя или пароль пользователя». В случае успешной авторизации должен быть осуществлен переход на основную страницу, содержащую простую таблицу, отображающую все записи из основной таблицы БД (все поля, кроме image и blob, для справочников - расшифровка). Таблица должна поддерживать сортировку по всем полям и пагинацию с количеством 20 записей на страницу. Также необходимо добавить возможность фильтрации данных в таблице по всем полям.

Для отображения таблицы на странице у авторизованного пользователя должен быть доступ на роль ROLE\_LIST\_VIEW, при ее наличии таблица видна, иначе - нет.

На стороне NodeJS сделать api «/api/data\_list», которая возвращает в формате JSON данные для отображения таблицы, с учетом сортировки, пагинации и фильтрации, заданных на клиенте. API доступна только, если у пользователя есть роль ROLE\_LIST\_VIEW. Сортировку, пагинацию и фильтрацию данных осуществлять на бэке или в БД.

В интерфейсе приложения разработать страницу для добавления/редактирования записей в основной таблице. Страница должна содержать все поля таблицы, расположенные вертикально, для справочников - выбор из выпадающего списка, для дат – поле с календарем, для вложений – выбор файла, для картинок – отображение ранее загруженной в виде превью с возможностью прикрепления другой картинки. Ряд полей сделать обязательными для заполнения по вашему выбору.

На странице с основной таблицей добавить кнопки «Добавить», «Изменить», «Удалить». Видимость (или доступность) кнопок определяется наличием у авторизованного пользователя доступа к ролям ROLE\_ADD, ROLE\_EDIT, ROLE\_DELETE соответственно. Кнопки «Добавить» и «Изменить» открывают страницу для добавления новой или для редактирования выделенной в списке записи. На стороне NodeJS сделать api «/api/get\_data» (получение данных для формы редактирования, доступ к апи по роли ROLE\_EDIT), «/api/edit\_data» (добавление/изменение записи, доступ к апи по роли ROLE\_EDIT), «/api/delete\_data» (удаление выбранной записи, доступ к апи по роли ROLE\_DELETE).

Результат работы передать в виде архива с исходными кодами, включающий также скрипт для создания и заполнения таблиц БД и инструкцию по установке и запуску приложения.