**Тестовое техническое задание**

на разработку мобильного приложения для администратора

**Требования**:

1. Приложение для администрирования булевых свойств объектов (абстрактно).
2. Разрабатывается на FLUTTER (с использованием BLOC).
3. Несколько пользователей (администраторов) работают с одним и тем же списком данных, время от времени меняя некоторые переключатели объектов.
4. Список объектов может дополняться (объекты могут добавляться) некой третьей стороной (оставим за рамками данного ТЗ, можно добавлять объекты через админку бэка).
5. Каждый объект имеет свойства: ИД, Наименование, Переключатель1, Переключатель2, Переключатель3, ПорядокСортировки (Число).
6. Список упорядочивается на клиенте по полю ПолеСортировки по возрастанию.
7. Каждый из пользователей может свободно включать или выключать переключатели прямо в списке, но не может менять другие свойства объектов, а также не может удалять объекты и/или добавлять их.
8. Изменения, внесенные другими пользователями и третьей стороной, должны отображаться в списке по мере обновления данных с сервера.
9. Организовать работу приложения в режиме оффлайн посредством локальной базы данных.
10. Синхронизация базы данных с сервером.
    1. В качестве имитации серверной части использовать сервис Firebase.
    2. При любых запросах к серверу делать имитацию серверных задержек случайной продолжительности от 0 до 5 секунд.
    3. Синхронизация должна происходить в фоне автоматически каждые 10 секунд.
    4. Сначала изменённые локально переключатели объектов должны менять состояние соответствующих переключателей объектов в базе сервера (выгрузка локальных изменений).
    5. Затем сразу локальная база должна прийти в соответствие с данными базы сервера (загрузка серверного состояния).
    6. При таком формате обмена может получиться так, что с момента выгрузки изменений на сервер и до момента получения данных с сервера, другое клиентское приложение уже успеет выгрузить на сервер свои изменения. Таким образом образуется коллизия. Предполагаем, что это нормальная ситуация, и всегда необходимо брать то состояние, которое на сервере в данный момент времени (то последнее загруженное изменение является верным).
    7. Случай 1. Два админа примерно одновременно изменили переключатель1 у одного и того же объекта, но второй админ изменил дважды (то есть вернул в исходное состояние). После этого оба админа примерно в одно и то же время начали синхронизацию. Приложение 1 (админ1) выгрузило изменения, сразу же приложение 2 выгрузило свои изменения, и в это время приложение 1 начало загружать данные по этому объекту. В таком случае в приложении админа 1 перключатель должен встать в состояние, которое успело образоваться на сервере (то есть вернуться в исходное состояние).
    8. Случай 2. Два админа примерно одновременно изменили РАЗНЫЕ переключатели (переключатель1 стал TRUE, переключатель2 стал FALSE) у ОДНОГО объекта. После этого оба админа примерно в одно и то же время начали синхронизацию. Приложение 1 (админ1) выгрузило изменения, сразу же приложение 2 выгрузило свои изменения. В таком случае изменения не должны затираться. Конечное состояние базы должно получиться: Переключатель 1 = TRUE, переключатель2 = FALSE, что должно соответствующим образом отразиться в локальных базах и затем в интерфейсе.
    9. Случай 3. При очередном обновлении данных, появился новый объект. Этот объект должен отобразиться в списке как все остальные объекты согласно его порядку сортировки (упорядочивания). То есть вероятна ситуация, когда объект появляется в середине списка. Это надо соблюсти.
    10. Если нет интернета, то синхронизация должна пройти тогда, когда интернет появится.
11. Каждые 2 секунды в фоне происходит обновление интерфейса из локальной базы.
12. При протягивании списка вниз должно начинаться принудительное обновление интерфейс, при этом должна отображаться “крутилка”.
13. Интерфейс списка должен соответствовать дизайну:   
    <https://www.figma.com/file/k8eQ6QhVpHYjXBmygotViT/Test-task?node-id=0%3A1>.
14. Использовать bloc 8 версия и выше
15. Покрыть тестами разработанный функционал.

Задание является на 100% тестовым, академическим, придуманным командой dev-stack.ru, и не является коммерческим. Все вероятные совпадения с элементами коммерческих заданий являются случайными.