План тестирования облачного сервиса и приложения по управлению роботом-пылесосом

1. **Unit – тесты:** Пишутся по мере написания кода на классы и методы API облачного сервиса, API обновления, авторизации и мобильного приложения.
2. **E2E-тестирование:** Проводим после завершения работы над кодом. Тестируем каждую систему на работоспособность. При этом воздействуем на систему через ее самые внешние интерфейсы и проверяем ожидаемую реакцию системы через эти же интерфейсы.

Таблица 1. Протокол тестирования API облачной системы

№ п/п	Вызываемый метод	Ожидаемый результат
1.1	checkAutorization()	Производится проверка введенных данных пользователя на соответствие с данными о логине и хэше пароля хранимыми в БД. Возвращает булево значение true при совпадении введенной информации.
1.2	createAuthorization()	Создание нового авторизированного пользователя с внесение полученных логина и хэша пароля в БД.
1.3	removeAuthorization()	Удаление данных авторизированного пользователя из БД.
2.1	addRobot()	Добавление нового робота-пылесоса в БД.
2.2	updateRobot()	Обновление хранимой в БД информации о роботепылесосе
2.3	removeRobot()	Удаление информации о роботе-пылесосе из БД
3.1	checkUpgradeRobot()	Проверка наличия обновлений программного обеспечения робота пылесоса. Возвращает булево значение true при наличии обновлений.
3.2	upgradeRobot()	Обновление программного обеспечения роботапылесос
4.1	setSequreGroup()	Добавление информации о группе пользователей допущенных к управлению устройством
4.2	createGroup()	Создание группы пользователей
4.3	updateGroup()	Обновление информации о группе пользователей
4.4	removeGroup()	Удаление группы пользователей
5.1	createSchedule()	Создание нового расписания уборки и запись в БД информации о нем.
5.2	updateSchedule()	Обновление информации о текущем расписании уборки в БД
5.3	removeSchedule()	Удаление текущего расписания уборки из БД

3. **UAT – тестирование:** готовый программный продукт тестирует ограниченный круг пользователей. Тестируется каждый case из UseCase диаграммы. При этом

группа людей изучает эффективность сервиса, его функционала. UAT нужен для того, чтобы понять: **а)** как ведет себя продукт в реальных условиях, соответствует ли результат задумке; **б)** выявить, были ли добавлены все возможные функции; **в)** проверить, есть ли ошибки, которые будут мешать пользователю.