Задача:

Создать тестовое новое приложение, в котором будут реализованы 5 сценариев:

1. Регистрация нового пользователя
2. Авторизация текущего пользователя
3. Проверка почты пользователя (при регистрации и авторизации)
4. Восстановление пароля
5. Изменение почты

Ссылки на макеты:

Прототип: https://xd.adobe.com/view/db52bdcb-d365-44c0-44b9-e0d121dff68d-ad63/
Develop: https://xd.adobe.com/spec/49c39483-d161-4281-7afe-55aee54100e0-3ff2/

Ссылка на иконки:
https://drive.google.com/drive/folders/1lEyd9bwyKo15kacB4kd4c51OULn1a8q6?usp=sharing

Минимальная версия iOS для проекта: 10

Пользователи сервиса делятся на категории и подкатегории:

- Зарегистрированные
    - С подтвержденной почтой
    - Почта не подтверждена
    - Почта отсутствует
- Незарегистрированные

После запуска загружается экран с 2мя кнопками:

- Зарегистрироваться
- Войти

Нажатие на “Зарегистрироваться” и сценарий регистрации:

- Заполнение обязательных полей информации о пользователе
    - Номер телефона (является логином)
    - Пароль
    - Фамилия
    - Имя
    - Почта

- Подтверждение согласия с правилами и условиями (отмечены цветом, никуда не ведут). Пока не отмечена галочка кнопка “отправить код” неактивна (см. макет).
- Отправление кода на указанный номер телефона (api/users/sms)
- Регистрация в системе с использованием полученного кода от пользователя (api/users/register).
    - Не обязательно, но будет приятно, если из пришедшего смс-сообщения код будет автоматически вставляться в приложение.
- Авторизация без участия пользователя с логином (номером телефона) и паролем, введенными на экране регистрации (api/auth/login) для получения необходимых параметров для следующего пункта
- Отправление письма на указанную почту (api/users/send-confirm)
- Подтверждение почты (параметр isConfirmed = true в запросе api/users/user)

Авторизация текущего пользователя и проверка подтверждения почты.

Процесс авторизации происходит по запросу api/auth/login
Проверка подтверждения почты осуществляется по параметру isConfirmed = true api/users/user

- isConfirmed=true

  Пользователь видит конечный экран.

- isConfirmed=false
    - Почта не подтверждена, но поле почты заполнено.
      Отправляется письмо с ссылкой для подтверждения почты (api/users/send-confirm)
    - Почта отсутствует
      Пользователю выдается сообщение с просьбой указать почту, по заполнению которой отправляется письмо с ссылкой для подтверждения почты.

Восстановление пароля

Пользователь вводит номер телефона, на который высылается смс с новым паролем. (api/users/restore)

Неверная почта

- Открывается UIAlertController с полем ввода, пустое поле ввода подтвердить нельзя (см. макет)
- Пользователь вводит новую почту
- Данная почта сохраняется в карточке клиента (api/users/change)
- При этом сервером сбрасывается статус подтверждения почты IsConfirmed, так что нужно заново отправить запрос на отправку письма (api/users/send-confirm)

Используемые API

Хост для запросов:
https://dev.musbooking.com Внимание! Это хостинг для отладки

Пример запроса:
https://dev.musbooking.com/api/categories/1/

Токен авторизации передается с префиксом “Bearer ”

POST дата в form-формате
postData="login=Test005&password=123456&name=test&surname=Test&phone=2132112312&email=Test005@mail.com&birthday=2014-05-17T12:03:42&"

Content-Type: application/x-www-form-urlencoded

Токен авторизации, получаемый при логине, передается в заголовке header :
Headers[‘Authorization’] = ‘Bearer ’ + token
По умолчанию авторизация требуется везде, если не указано обратное

Формат результатов: JSON
Все названия полей camelNames, с маленькой буквы (стандарт для JS Json)

Регистрация нового пользователя

Регистрация нового пользователя мобильного приложения
Авторизация не требуется

Внимание! Регистрация происходит в 2 этапа:

- Пользователю отсылается смс уведомление (в мобильном приложении на форме регистрации по кнопке)
- Пользователь при регистрации указывает присланный код, который он получил в СМС сообщении

Особенности:

- Код кэшируется в памяти сервера
- Код не имеет срока действия
- Привязка кода только к номеру телефона
- Авторизация не требуется

Отправка кода

URL: (api/users/sms)
Метод: POST
Параметры запроса: все параметры обязательны:
phone (String): Телефон, формат +7 XXXXX, сервер сам нормализует
Ответ:
При удачной отправке СМС: 200 OK,
При наличии пользователя в базе с таким номером телефона - 400 ошибка с соотв текстом
При неудачной или ошибке 400 Bad Request, и текст ошибки (который выдается СМС-агрегатором)

Собственно регистрация

URL: (api/users/register)
Метод: POST
Параметры запроса:
login (String) минимум 6 знаков: Логин (опционально)
password (String) минимум 6 знаков: Пароль (опционально, иначе генерируется)
name (String): Имя
lastname (String): Фамилия
surname (String): Отчество
phone (String): Телефон (обязательно!!)
code (String): Код подтверждения авторизации из СМС
email (String): Почта
birthday (datetime): 22.01.2015 15:23:56: День рождения
Ответ:
При удачной регистрации: 200 OK, Если клиент уже есть в базе, то его данные не изменяются, создается только логин
При неудачной или ошибке 400 Bad Request, и текст ошибки
Пароль проверяется на сложность (от 6 символов, + числа, +спецсимволы)
Если есть такой клиент, то привязываем аккаунт к клиенту

Вход пользователя

Процедура входа пользователя мобильного приложения

URL: api/auth/login
Метод: POST
Параметры запроса: все параметры обязательны:
login (String) минимум 6 знаков: Логин
password (String) минимум 6 знаков: Пароль
Ответ:
token: “xxxx”
При удачной попытке входа: 200 OK (ответ содержит токен, который далее приложение должно использовать для запросов ко всем функциям требующим аутентификации пользователя вместо логина и пароля.
Токен действителен только одну сессию и максимум в течении 24 часов).
При неудачной или ошибке 400 Bad Request

Запрос личных данных пользователя

Запрос данных пользователя мобильного приложения

URL: api/users/user
Метод: GET
Параметры запроса:
Ответ:
При удачном запросе: 200 OK, и данные пользователя (см. пример).При неудачной или ошибке 400 Bad Request (в т.ч. в случае просроченного токена)

id (guid): ID пользователя
login (String) минимум 6 знаков: Логин
name (String): Логин
firstName (string): Имя
lastName (string): Фамилия
surName (String): Отчество
phone (String): Телефон
email (String): Почта
birthday (String) 22.01.2015 15:23:56: День рождения
room: название последней забронированной комнаты
base: название последней забронированной базы
isConfirmed (bool): подтверждение почты
fcm (string): Firebase Cloud Messanging Token
parts: [] - массив значений по партнерским зонам
domainId - id партнерской зоны
name - название партнерской зоны
forfeit - сумма штрафов по партнерской зоне
isBanned: bool - забанен или нет

discount (int) - скидка по партнеру

Изменение личных данных

Изменения в личном кабинете пользователя мобильного приложения

URL: api/users/change
Метод: POST
Параметры запроса: параметры необязательны, т.е. какой параметр будет прислан тот и будет сохранен:
name (String): имя
lastname (String): Фамилия
surname (String): Отчество
*** phone (String): телефон - убрано изменение телефона, тк логин
email (String): почта (при этом сбрасывается статус подтверждения почты IsConfirmed)
birthday (String): дата рождения
fcm (string): Firebase Cloud Messanging Token

Ответ:
При удачном запросе: 200 OK.При неудачной или ошибке 400 Bad Request

Восстановление пароля

Восстановление пароля на почту или смс
Поиск пользователя по login или phone или mail

URL: api/users/restore
Метод: POST
Параметры запроса:
login (string) - логин пользователя
mail (string) - либо почтовый ящик
phone (string) - либо номер телефона в формате XXX (все спецсимволы вырезаются)
sms (bool) - если задан true, то пароль восстанавливается по SMS, иначе высылается на email

Ответ:
200 (Ок), если все прошло нормально

Отправка email

Отправка почтового подтверждения клиенту
Ссылка для подтверждения (пример): {http://xxxx}/api/users/confirm?id={User.Id}

URL: api/users/send-confirm
Метод: POST
Параметры запроса:
Обязательно токен! Почта посылается текущему пользователю

Ответ:
“ok”, 200 (если успешно).BadRequest (если ошибка)