ТЗ на ЛК и мобильное приложение для химчистки

01/04/2016

1 Описание проекта

Рабочее название проекта - cleanmeupscotty. Состоит из двух частей - личного кабинета пользователя (представляет собой веб-сайт на Битриксе) и двух мобильных приложений (или одним с разными доступами и разным функционалом).

1.1 Личный кабинет (сайт)

Суть заключается в следующем - в личный кабинет (ЛК) на сайт поступает информация от сервера, который обрабатывает данные ПО химчистки. Например о статусе определенной вещи. Также видна некая статистика - сколько вещей было сдано, сколько денег потрачено, сколько бонусных баллов заработано. Сайт также отдает информацию на сервер химчистки.

1.2 Мобильное приложение для клиента

Дублирует основной функционал личного кабинета (выводит данные сервера химчистки). Дополнительно позволяет вызывать водителя к текущему местоположению для того, что бы он забрал вещи.

1.3 Мобильное приложение для водителя

Позволяет принимать заказ от клиента. Отправляет данные на сервер химчистки.

2 Понятия

- Сайт, личный кабинет, ЛК - веб-сайт, размещенный на хостинге, основанный на системе 1С Битрикс
- Сервер химчистки - физический сервер, на котором крутится определенное ПО, которое использует химчистка. Сервер собирает данные с компьютеров каждой точки и центрально их хранит. Условно можно сказать - ПО точек это интерфейс, сервер - это база данных.
- Мобильное приложение, МП - мобильное приложение для сайта, которое дублирует его функции.

3 Схема работы

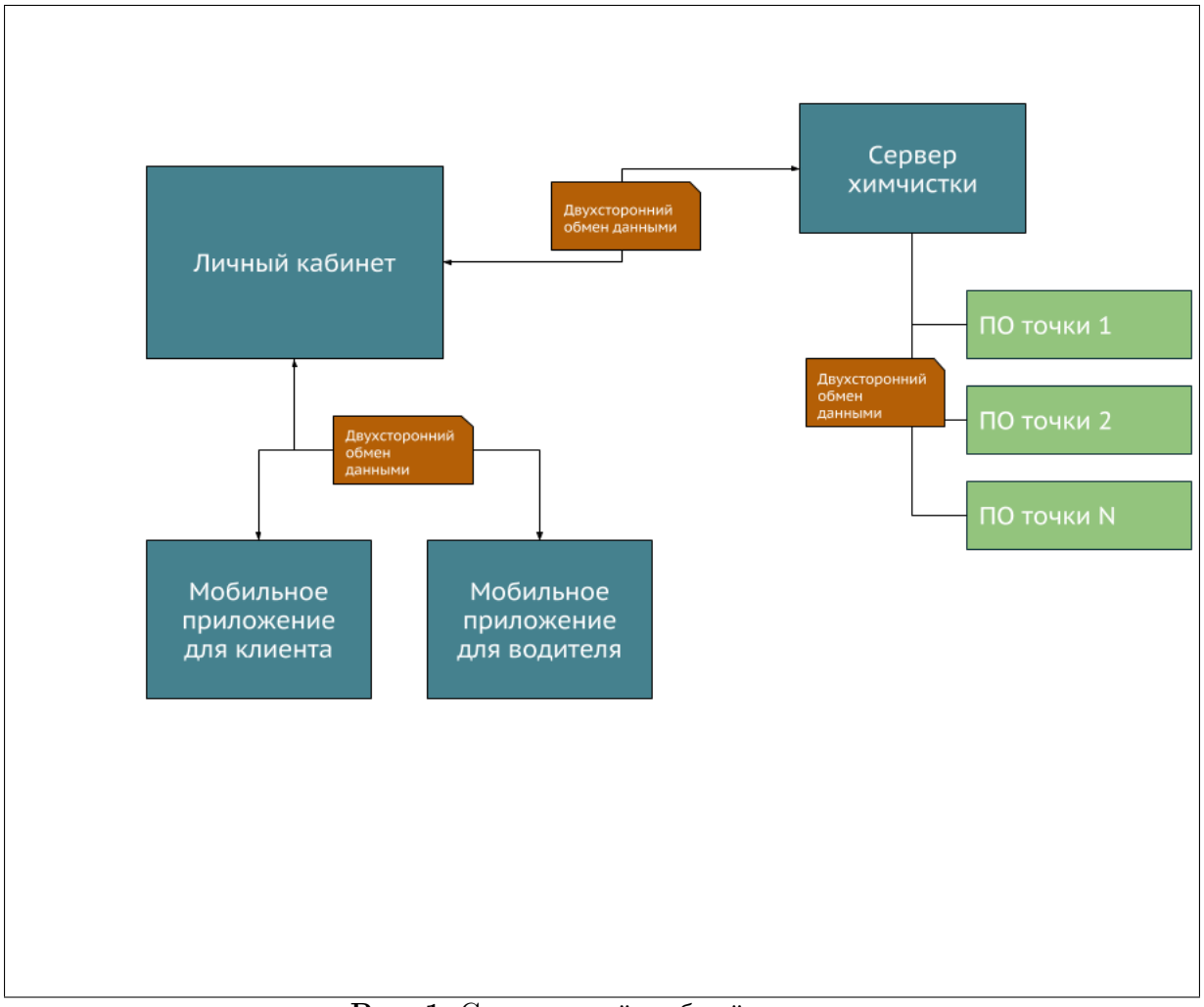

Рис. 1: Схема связей в общей системе

Из точек приема 1...N поступают данные о заказах на общий сервер. Сервер отправляет данные также и обратно (эта сторона не касается проекта, данный обмен налажен на стороне ПО химчистки).

У сервера есть интерфейс приложения (application interface, API), с помощью которого можно запрашивать данные с сервера. Эти данные показываются клиенту в его личном кабинете (на сайте или в мобильном приложении).

Заказывая сбор вещей на дом, клиент вызывает водителя, который забирает грязные вещи. Водитель через свое приложение отмечает что он их принял. Клиент в свою очередь отмечает, что он их отдал (нужно обязательно).

4 Параметры

4.1 Архитектура проекта

Практически все данные проект принимает, обрабатывает и отправляет на сервер химчистки. На своей стороне хранится самый необходимый минимум.

Проект изначально должен соблюдать правила объектно ориентированного программирования. Т.е. любой пользователь должен быть объектом класса "пользователь". Это распространяется на все сущности. Все действия системы должны быть методами своих классов. Псевдокод:

```
#Class description

class User < ActiveRecord::Base

def initialize(name, phone, mail)
        @name = name
        @phone = phone
        @mail = mail
end

# Register method
def register(user_params)
        User.create(user_params)
end

# Allow only whitelisted params
private
        def user_params
                params.require(:user).permit(:name, :phone, :mail)
        end
```

Программирование должно осуществляться по подходу TDD - test driven development. Это означает, что перед написанием функционала необходимо написать тест (который заведомо провалится, так как функционала нет). Далее пишется минимальный функционал, который позволяет пройти данный тест.

Псевдокод теста:

```
require 'user'

Rspec.describe User, "#register" do
  context "when␣user␣is␣completely␣new" do
        it "creates␣a␣new␣entry␣with␣valid␣data" do
          user = User.new
      user.name = "Vasya␣Pupkin"
          user.phone = "8␣903␣227␣88␣74"
          user.mail = "vasya@pupkin.ru"
```

```
          user.register

          expect(User.count).to change{User.count}.by(1)
        end
    it "does not create a new entry with invalid data" do
      user = User.new
          user.name = "Vasya Pupkin"
          user.phone = "j1lkh3i1u2y98"
          user.mail = "someinvalidtext"
          user.register

      expect(User.count).not_to change{User.count}.by(1)
        end
  end
end
```

T.e. в данном случае мы тестируем создание нового пользователя с валидными данными (первый случай) и с невалидными данными. В первом случае мы ожидаем увеличение общего кол-ва пользователей на 1, во втором случае мы не ожидаем увеличений.

4.2 Сущности системы

- Аккаунт пользователя (пользователь)
- Прайс-лист
- Квитанция
- Изделие
- Скидка

4.2.1 Структура сущностей

К лицевому счету (аккаунту) привязаны квитанции (одна или несколько). К квитанциям привязаны изделия (одно или несколько). Эти данные не редактируются на сайте, а подгружаются как есть с сервера.

4.2.1.1 Товары

Изделия группируются также в свою очередь. Уровней 3:

- Раздел - принадлежность к типу химической обработки, например "хим.чистка основной группы". Может быть "хим. чистка кожаных изделий".
- Категория - общая группа товаров, например "костюмная группа"
- Товар - конкретный товар, например "пиджак"

4.2.1.2 Прайслист Прайслист - есть один (по умолчанию). К нему привязаны, если нет иного, все изделия. Если к изделию привязывается другой прайс-лист, то он превалирует на дефолтным. Данные прайс-листа подгружаем с сервера, т.е. мы должны только контролировать какой сейчас актуальный прайс-лист для определенного товара.

Пример - дефолтно стоимость чистки шелковой сорочки составляет 490 рублей. Мы хотим провести акцию "Сорочки по 200! для этого создается новый прайс-лист с ограниченным сроком действия и для товаров категории "сорочки"прописываем цену "200 руб". Теперь на время действия прайслиста цена чистки любой сорочки должна быть 200 рублей.

4.2.1.3 Скидки Скидки не должны суммироваться. Т.е. если у клиента есть 5% скидка и активен прайс-лист, по которому сорочки чистятся за 200 рублей, то данный клиент также платит 200 рублей. Иными словами личные скидки распространяются только на дефолтный прайслист.

4.3 Личный кабинет

В личном кабинете клиент может зарегистрироваться или авторизоваться, если его учетная запись уже есть в системе.

4.3.1 Авторизация

Если клиент уже есть в системе, то при первичной авторизации, нужно предложить ему ввести номер телефона и подтвердить смс-сообщением, после авторизации сразу предложить сменить пароль и дальнейшую авторизацию он может осуществлять с помощью логина и пароля.

4.3.2 Регистрация

Для регистрации клиенту необходимо внести только телефон. На данный номер приходит смс с паролем, который можно скопировать. Клиент может быть уже зарегистрирован в системе (так как сдавал вещи в точке приема и при этом оставил свой номер телефона).

При первичной регистрации (проверяется по отсутствию телефона в базе сервера) схема должна работать следующим образом - клиент заполняет фио, телефон и почту, эти данные отправляются на сервер. Сервер создает у себя новую учетную запись и дополняет её своими полями. Далее сайт забирает уже полную карточку клиента.

4.3.2.1 Поля

- ФИО - отдельные поля
- email
- телефон
- адрес
- лицевой счет

- класс клиента - содержит информацию о скидке клиента (например класс 12 означает скидку в 3%)
- идентификатор - равен номеру дисконтной карты (не является ID в системе)
- тип идентификатора - может быть магнитная карта или промокод
- тип лицевого счета
- прайслист - привязка к сущности "прайслист". У клиента может быть только один
- аватарка - с функцией кропа под размер сайта
- дата рождения
- тип клиента (юрик/физик)

Так эти поля выглядят в ПО химчистки:

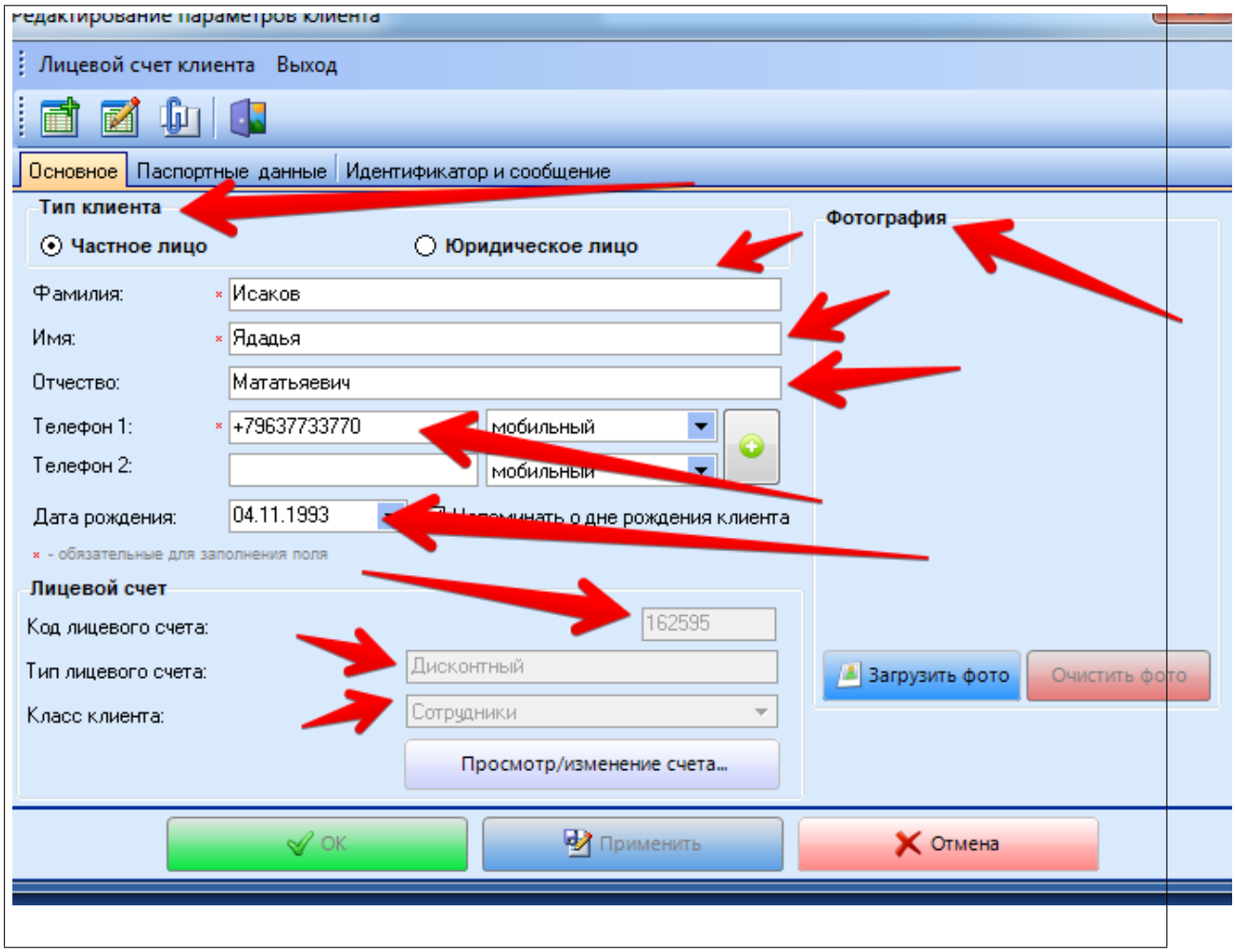

Рис. 2: Обзор полей в ПО химчистки

4.3.2.2 Доступ Пользователь сайта/мобильного приложения может менять только следующие данные:

- Телефон
- ФИО
- Почту
- Адрес
- Фото

4.3.3 Запросы

Личный кабинет и сервер обмениваются данными каждые 30 минут. В идеале мы отдаем и принимаем только изменения и делаем это в режиме реального времени. Но если можно передавать только целиком скоп данных, то делаем это каждые 30 минут.

4.3.3.1 Примеры

- Клиент регистрируется на сайте -> информация попадает на сервер ПО химчистки
- Клиент приходит в точку в городе, сдает одежду, его регистрируют в ПО конкретной точки. Эти данные попадают на сервер, мы их оттуда забираем. Клиент в итоге оказывается в системе сайта.
- Клиент меняет номер телефона на сайте -> отсылаем информацию на сервер
- Клиент приходит в точку приема и просит поменять телефон -> мы забираем обновившиеся данные с сервера

4.3.3.2 POST Request На сервер химчистки отправляется POST-запрос, который меняет данные на сервере. Передаем сущность, которую хотим поменять, идентификатор сущности и значения.

$$POSThttp://api.cleanmeupscotty.ru/params?client\&id = 7817236178\&name = Oleg$$

4.3.3.3 GET Request К серверу химчистки отправляется GET-запрос, который содержит в себе сущность, к которой обращаются и уникальный идентификатор сущности. Пример запроса:

$$GEThttp://api.cleanmeupscotty.ru/params?client\&id = 7817236178$$

4.3.4 Ответ

От сервера химчистки приходят данные в формате XML или JSON. Перечень данных:

- Кол-во вещей в обработке
- Общее кол-во сданных вещей
- Накопленные баллы
- Статус клиента

- Перечень квитанций (статус квитанции, сумма по квитанции, дата оплаты и прочие свойства квитанций)
- Перечень вещей в обработке (для каждой вещи получаем уникальный айди от химчистки, наименование, статус по химчистке, цену на
- Общий перечень всего сданных вещей
- Вся статичная информация клиента (фио, телефон, почта и прочее)
- Прайслисты

Пример ответа может выглядеть так:

```
{"client": {
  "id": "7817236178",
  "status": "vip",
  "current_items": 3,
  "bonus_miles": 9351,
  "clothes_current": {
    "cloth": [
      {"id": 4125124, "value": "Platje", "status": "clean", "price
         ": 490},
      {"id": 89172387, "value": "Rubashka", "status": "being
         cleaned", "price": 199},
      {"id": 171273, "value": "Kostjum", "status": "preparation",
         "price": 2500}
    ],
    "furniture": [
        {"id": 7887182, "value" "Pokrivalo on divana", "status": "
           preparation", "price": 4900}
    ]
  },
  "clother_all": {
        "listing_all_clothes": ...
  }
}}
```

4.3.5 Вызов водителя

Аналогично мобильному приложению (см. пункт 4.4.1.1)

4.4 Мобильное приложение клиента

Мобильное приложение является полным дублем веб-личного кабинета и показывает в удобном интерфейсе те же данные (см. пункт 4.3.4). Дополнительно мобильное приложение позволяет вызвать водителя для сбора вещей, которые необходимо отдать в чистку.

4.4.1 Запросы

Мобильное приложение основано на веб-личном кабинете, соответственно мобильное приложение "общается"с сайтом, а тот, в свою очередь с сервером ПО химчистки. Дополнительно к общему функционалу у мобильного приложение есть уникальный функционал - заказ водителя.

4.4.1.1 Вызов водителя При нажатии на кнопку у пользователя есть выбор двух опций - время и место. Время по умолчанию стоит "сейчас". Пользователь может альтернативно выбрать определенное время. Место стоит по умолчанию "здесь"и выбирает текущее местоположение пользователя, альтернативно пользователь может выбрать местоположение.

Для местоположения - так как сервис рассчитан на Москву выбираем только улицу, дом и квартиру.

4.4.1.2 Подтверждение заказа После того, как водитель у себя ввел все данные забранных вещей мы должны получить от клиента подтверждение, что все окей. Предложение из двух вариантов - либо в приложение водителя делаем экран, где клиент может расписаться, либо в приложении клиента после того, как водитель ввел все данные выводим кнопку "подтвердить заказ". Здесь на усмотрение разработчика.

4.4.1.3 Оплата При создании новой квитанции (это означает, что вещи были приняты в точке приема) необходимо создавать пуш-уведомление. По переходу показываем экран с суммой заказа (и на развороте перечень вещей) и кнопкой "оплатить". Если у клиента уже прикреплена карта, то списываем с нее сумму денег. Если нет, то переадресовываем на экран, где он эти данные должен ввести.

Очень важный момент - при создании новой квитанции на стороне сайта проверяем обязательно, если ли корреспондирующий заказ у водителя. Так как квитанцию можно создать и при приходи клиента в точку приема лично (и он таким бы образом оплатил заказ дважды).

Если корреспондирующий заказ есть, то все работает по вышеописанной схеме (пуш-уведомление, которое ведет на страницу оплаты), если "водительского"заказа нет, то вообще ничего не происходит.

Также в мобильном приложении важно иметь список квитанций со статусами оплачен/не оплачен, что бы в случае потери пуш-уведомления у клиента все равно была возможность оплатить квитанцию.

4.5 Мобильное приложение водителя

Данное приложение получает заявки от мобильных приложений клиентов.

4.5.1 Схема работы

Водитель может принять заказа или проигнорировать его. После того, как он принял заказ должен открываться навигатор и показывать маршрут к месту назначения клиента. При-

нятые вызовы выводим списком и сортируем по дате, на которую вызвали. Т.е. если кто-то вызвал водителя на завтра, этот заказ будет в списке более ниже, чем тот, где вызвали на сегодня.

4.5.1.1 Подтверждение забора вещей По прибытии и забора вещей водитель должен отметить в системе, что он забрал вещи (или не забрал, в этом случае нужно указать причину) и какие вещи он забрал. Для этого ему необходимо вывести простой список вещей (не такой, как в пункте 4.2.1.1! Более простой, т.е. например "пиджак "сорочка "брюки а не "пиджак замшевый "пиджак шерстяной"и т.д.)

Данный лист мы не подтягиваем с сервера химчистки, этот лист является неким дигитальным вариантом обычного бумажного описного листа. Этот лист также никуда не передаем, просто храним у себя на стороне Битрикса.

Если навигатор отдал назад ответ, что водитель приехал и он не нажал ни одну из кнопок, то экран должен мигать красным.

4.5.1.2 Прием заказа Заказ попадает ко всем водителям сразу.

4.5.1.3 Навигация Имеет смысл использовать стороннюю, гугл или яндекс навигаторы например. От них необходим ответ, что по GPS координатам дивайс прибыл к цели.