

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, чтобы 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

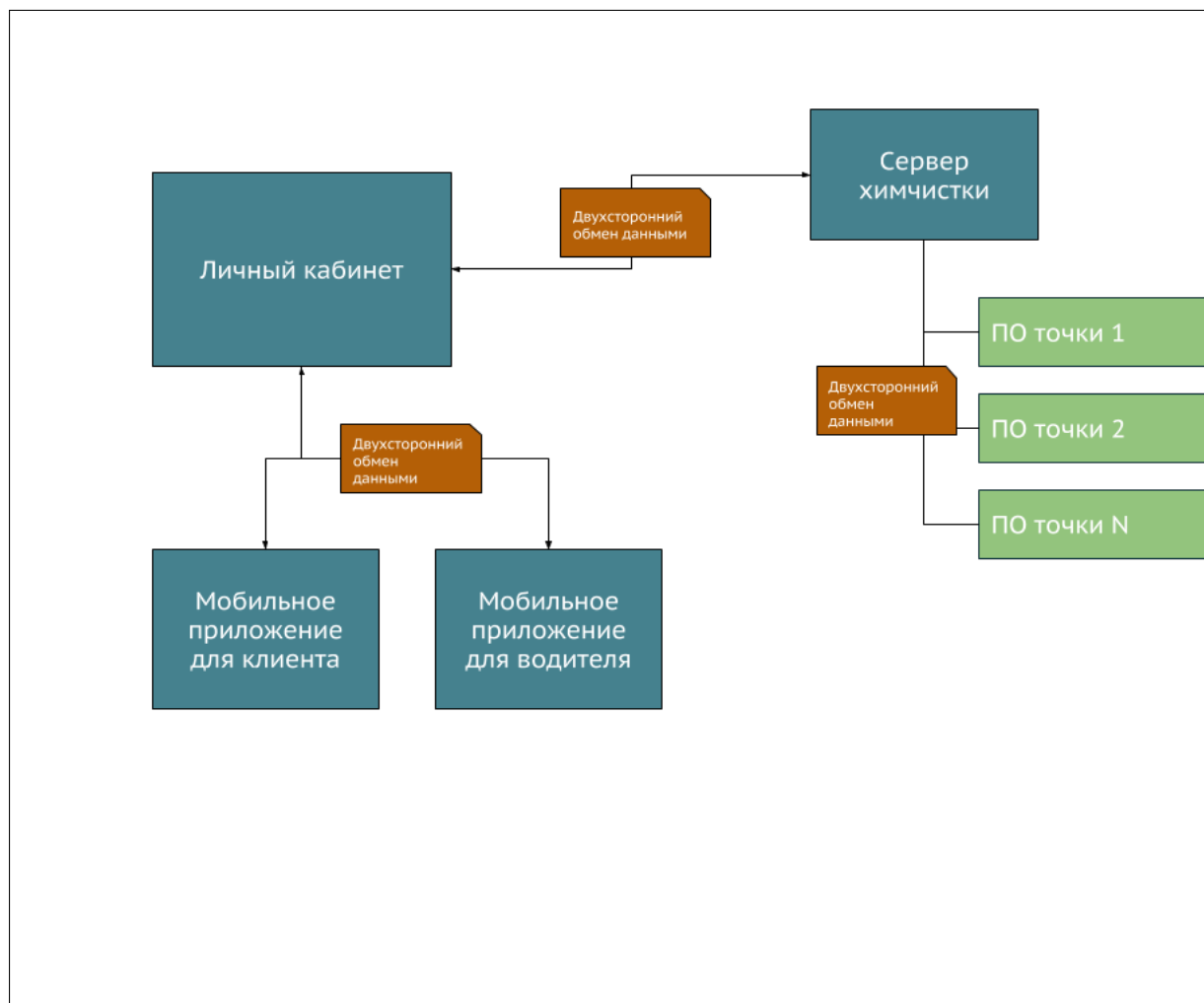


Рис. 1: Схема связей в общей системе

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 хмчистки. 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
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 = "89032278874"
      user.mail = "vasya@pupkin.ru"
    end
  end
end
```

```

        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

```

Т.е. 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
- Разделы
- Категории
- Товары
- Классы

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 - некая сущность (табличного вида) ПО химчистки, в которой содержатся цены на те или иные позиции. На уровне ПО химчистки может быть привязан как к точке приема, так и к конкретному клиенту.

Данные прайс-листа подгружаем с сервера, т.е. мы получаем всегда актуальный прайс-лист, который действует для станции, идентификатор которой мы передаем в запросе (если у клиента свой собственный прайс-лист, то он превагирует над прайс-листом станции)

4.2.1.3 Квитанции Все квитанции имеют следующий номер - XX / YYYYYY, где XX - это идентификатор станции. Соответственно на основе XX и идет деление квитанций - на стороне химчистки есть только одна станция, отвечающая за мобильное приложение. Именно по номеру станции мы должны определять, является ли данная квитанция мобильной или ритейловской.

4.2.1.4 Классы (Ранее - скидки) Никакой смысловой нагрузки не несущая сущность. На уровне ПО химчистки в классах хранятся текстовые описания скидок. Храним их у себя и выводим в ЛК клиента.

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 ф.и.о., 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 - по маске. При смене клиент должен подтвердить новую почту.
- телефон - по маске. При смене клиент должен подтвердить новый 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
- код лицевого счета - устанавливается на уровне magicsoft'a (является айдишником magicsoft'a)
- прайслист - привязка к сущности "прайслист". У клиента может быть только один
- аватарка - с функцией кропа под размер 300x450 и только формат jpg (важно, так как ПО химчистки не принимает другие форматы)
- дата рождения - по маске
- тип клиента (юриск/физик)

Так эти поля выглядят в ПО химчистки:

Редактирование параметров клиента

Лицевой счет клиента Выход

Основное Паспортные данные Идентификатор и сообщение

Тип клиента

☒ Частное лицо ☐ Юридическое лицо

Фамилия: * Исаков

Имя: * Ядадя

Отчество: Мататьяевич

Телефон 1: * +79637733770 мобильный

Телефон 2: мобильный

Дата рождения: 04.11.1993

☐ Напоминать о дне рождения клиента

* - обязательные для заполнения поля

Лицевой счет

Код лицевого счета: 162595

Тип лицевого счета: Дисконтный

Класс клиента: Сотрудники

Просмотр/изменение счета...

Фотография

Загрузить фото Очистить фото

OK Применить Отмена

Рис. 2: Обзор полей в ПО химчистки

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

- Телефон
- Почту
- Адрес
- Фото

4.3.3 Подтверждение почты

Если клиент уже есть в базе ПО химчистки и там внесен email адрес (например клиент в точке просто устно продиктовал), то в ЛК напротив такого адреса нужно поставить знак ошибки и что данный адрес нужно подтвердить (также в шапке нужно показать, аналогично с порталом битрикса, что есть 1 уведомление). Как только клиент подтверждает мэйл, то уведомление нужно скрыть вместе со знаком ошибки.

4.3.4 Запросы

Личный кабинет и сервер обмениваются данными в режиме реального времени. Для этого на стороне ПО химчистки настроен веб-сервер, который может принимать GET и POST запросы, а также сам инициировать POST запросы. Полный перечень запросов ниже.

4.3.4.1 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

4.3.4.2 POST Request от мобильного приложения

4.3.4.2.1 [U+FFFD] данные пользователя На сервер химчистки отправляется POST-запрос, который меняет данные на сервере. Передаем сущность, которую хотим поменять, идентификатор сущности и значения.

Параметры запроса:

- $first_{name} - surname -$
- $fathers_{name} - email -$
- address - место проживания пользователя
- picture - фотография пользователя

Пример:

POSThttp://api.mymagicsoft.suzy-wong.ru/params?client&id=7817236178&name=Oleg

Данные запрос поменяет имя клиента с id 7817236178 с текущего на "Олег".

4.3.4.2.2 [U+FFFD] данные о квитанции На сервер химчистки отправляется POST запрос, который меняет данные о квитанции. Конкретно - меняется только статус оплаты (оплачен/неоплачен).

Пример:

POSThttp://api.mymagicsoft.suzy-wong.ru/params?receipt&id=25-567162&payed=true

Данные запрос поменяет статус квитанции с id 567162 терминала 25 на "оплачено".

4.3.4.3 POST Request от сервера химчистки Сервер ПО химчистки отправляет POST запросы на веб-сервер мобильного приложения при изменениях в своих сущностях. Перечень сущностей:

- Данные пользователя (перечислены в пункте 4.3.2.1
- Прайслист
- Квитанция

4.3.4.3.1 [U+FFFD] данные пользователя При изменении данных пользователя (или создании новых) в ПО MagicSoft на сервер мобильного приложения приходит POST запрос, который меняет эти же данные на сервере мобильного приложения.

Пример:

POSThttp://api.cleanmeupscotty.ru/params?client&id=7817236178&phone=+79689032278874status=

В данном случае на стороне ПО химчистки у клиента поменялся телефон, его статус и адрес электронной почты.

4.3.4.3.2 [U+FFFD] райслист

При изменении данных в прайсliste/прайслистах (или при создании новых) ПО химчистки отправляет POST запрос на сервер личного кабинета.

Пример:

POSThttp://api.cleanmeupscotty.ru/params?pricelist&id=926813&category=kosjumnaja_group_pacloth

В данном запросе для прайслиста с идентификатором 926813 поменялась цена на пиджак в костюмной группе. Новая цена стала 2500 рублей.

4.3.4.3.3 [U+FFFD] витанции

При изменении данных квитанций (или созданий новых) ПО химчистки отправляет POST запрос на сервер личного кабинета.

Пример:

POSThttp : //api.cleanmeupscotty.ru/params?receipt&id = 25-567163&clothes = [kostjum, rubashka, pid]

В данном примере создается новая квитанция на стороне ПО химчистки и через POST запрос к серверу личного кабинета на последнем также создается данная квитанция.

4.3.4.4 GET Request К серверу химчистки отправляется GET-запрос, который содержит в себе сущность, к которой обращаются и уникальный идентификатор сущности. Запрос рассчитан в первую очередь для первичной инициализации базы. Пример запроса для получения данных всех клиентов:

GEThttp : //api.mymagicsoft.suzy - wong.ru/params?clients

Пример запроса для получения данных прайслистов:

GEThttp : //api.mymagicsoft.suzy - wong.ru/params?pricelists

Пример запроса для получения данных квитанций:

GEThttp : //api.mymagicsoft.suzy - wong.ru/params?receipts

4.3.5 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 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
- Перечень квитанций (статус квитанции, сумма по квитанции, дата оплаты и прочие свойства квитанций)

Пример ответа для пользователей может выглядеть так (JSON):

```

1 {"clients": {
2   "id": "7817236178",
3   "status": "vip",
4   "bonus_miles": 9351,
5   "first[U+FFFF] [U+FFFF] [U+FFFF]
6 [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF]
7 [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF]
8   "email": "oleg@olegovich.ru"
9 }, {
10   "id": "512412312",
11   "status": "peasant",
12   "bonus_miles": 256,
13   "first[U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF]
14 [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF]
15 [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF] [U+FFFF]
16   "email": "alina@olegovna.ru"
17 }, {...}
18 }

```

Пример ответа для прайслистов может выглядеть так (JSON):

```

1 {"pricelists": {
2   "categories": {
3     "subcategories": {
4       "id": "8192",
5       "category": "himchistka_osnovnoy_gruppi",
6       "subcategory": "kostymnaya_gruppa",
7       "clothes": {
8         "pidzhak": 5400,
9         "rubashka": 1200,
10        "brjuki": 3100
11      },
12      "id": "1123",
13      "category": "himchistka_osnovnoy_gruppi",
14      "subcategory": "kozhanaja_gruppa",
15      "clothes": {
16        "kurtka": 12400,
17        "perchatki": 350,
18        "shtani": 4800
19      },
20      {...}
21    },
22    {...}
23  }
24 }

```

Пример ответа для квитанций может выглядеть так (JSON):

```

1 {"receipts": {

```

```

2      "station": 25,
3      "id": 142123,
4      "total_price": 21500,
5      "created_at": 110520161914,
6      "client_id": 8198289,
7      "clothes": {
8          "pidzhak": 3400,
9          "rubashka": 1200,
10         "shtori": 17500
11     },
12     "station": 12,
13     "id": 781729,
14     "total_price": 700,
15     "created_at": 110520161914,
16     "client_id": 8198289,
17     "clothes": {
18         "perchatki": 300,
19         "trusi": 400,
20     }
21 }

```

4.3.6 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 4.3.5). 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 приложения 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 okay. 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 идентификатор станции (приемного пункта). На стороне химчистки будет отдельная станция для доставки. Только если квитанция возвращается и в ответе видим именно необходимый идентификатор, то в таком случае отправляем пуш-уведомление и списываем деньги с клиента. (Так как квитанцию можно создать и при приходи клиента в точку приема лично (и он таким бы образом оплатил заказ дважды)). Такой квитанции также нужно проставлять категорию (= "доставка"), остальные квитанции с точек будут иметь иную категорию (например "ритейл")

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

Данные об оплате должны передаваться на сервер химчистки с помощью установки "галки" об оплате.

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. Принятые 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.