ТЗ VACLS (v0.0.1)

**Аннотация:**

Программный комплекс VACLS предназначен для идентификации уже посещавших магазин клиентов. Таким образом она избавляет от необходимости выдавать клиентам карты лояльности или скидочные купоны. Когда клиент подходит к кассе, камера делает захват изображения и отправляет его на сервер для распознавания. Если сервер узнает клиента, то он отправит кассиру сообщение с текстом о том, что нужно делать дальше. С помощью приложения администрирования можно управлять реакциями на каждую конкретную ситуацию, например повышать скидку каждое двадцатое посещение и т.п. Реакция должна быть описана на языке JavaScript. Кроме того административное приложение выведет статистику по посещениям и по выданным скидкам, можно будет увидеть, сколько новых людей приходит.

Проект функционально должен состоять из следующих модулей (требования для каждого формулируются отдельно, затем описывается общее взаимодействие модулей):

1. **Клиент-модуль на кассе**
2. **Серверное приложение**
3. **Приложение для администрирования**

**1. Клиент-модуль на кассе**

**Назначение модуля:**

Отслеживание движения в области кассы с помощью веб-камеры и при наличии лица покупателя захват изображения в случае подходящего ракурса и отправка его на сервер для распознавания. После того, как сервер распознает изображение, вывести на экран окно с инструкцией для кассира, инструкция получается так же с сервера. По закрытию окна необходимо продолжить отслеживание движений и лиц.

**Функциональные требования:**

1. Активировать определение наличия лица в кадре при наличии движения в кадре (Для этого непрерывно читается поток с веб-камеры)
2. При нахождении подходящего лица в кадре (Лицо смотрит примерно в камеру, ничем не перегорожено, не пересвечено и не недосвечено, не имеет посторонних предметов, таких как темные очки или паранджа) делает захват изображения и отправляет его на сервер для распознавания. Формат запроса описан в требованиях к выходным данным.
3. После приема ответа на запрос выводится окно с инструкциями для кассира.
4. По закрытии окна с инструкциями продолжается чтение потока камеры

**Требования к выходным данным:**

1. Запрос на распознавание подается с помощью протокола https методом post и содержит json-строку со следующими полями:
   1. Base64 - изображение
   2. id кассы
   3. token для разрешения доступа к api сервера
   4. время захвата снимка
2. JavaFX окно с сообщением для кассира, полученным от сервера.
3. Информативный лог-файл

**Требования к входным данным:**

1. Файл с параметрами передается в качестве консольного параметра и содержит
   1. Источник потока видео.
   2. Куда писать лог
   3. В каком разрешении делать снимок
   4. Какие горячие клавиши использовать для закрытия окна и принудительного снимка
   5. Путь к библиотеке OpenCV
   6. Адрес и порт сервера
   7. Максимальное время ожидания ответа от сервера
   8. ID кассы
2. В качестве ответа сервера получаем JSON - строку со следующими полями:
   1. Сообщение кассиру
   2. Код ошибки (0 - если ошибок нет, 1 - ошибка авторизации, 2 - ошибка распознавания, 3+ - другая ошибка)
3. Поток с видеокамеры
4. Горячие клавиши клавиатуры

**Требования к окружению:**

1. OC MacOSX/Linux/Windows, JRE 8 +, OpenCV 3.2.0 Native Build
2. Веб-камера с разрешением 720p

**2. Серверное приложение**

**2.1. Требования к составу выполняемых функций**

Программа должна обладать следующим функционалом:

1. Получение фотографии и другой необходимой информации для идентификации клиента от кассового клиент-модуля.
2. Возможность конфигурации сервера, клиент-серверной связи и БД для хранения полученной и обработанной информации посредством административного приложения.
3. Идентификация клиента по полученной от кассового модуля фотографии и соотнесение с имеющимися в БД обработанными масками лиц посредством библиотеки OpenCV.
4. Запись/обновление обработанной маски лица (обезличенной) с необходимой служебной информацией в БД.
5. Передача установленной администратором информации (скидок, акций, и тд.) кассовому модулю.
6. Передача администратору собранной статистической информации за определенные промежутки времени и другим параметрам.

**2.2. Требования к организации входных данных**

Серверное приложение принимает параметры работы при запуске из конфигурационного файла, задаваемого админ. приложением или вручную. Конфиг содержит следующие поля:

1. Порт для передачи данных на кассу
2. Порт для администрирования системы лояльности клиентов (прописывание скриптов и скидок и тд.)
3. Путь для записи лог-файла
4. Максимальное число потоков-обработчиков запросов

При работе сервер принимает https запросы от касс и администрирующего приложения (post и get) и обрабатывает согласно функциональным требованиям.

От кассового клиентского модуля данные принимаются в JSON формате согласно его выходным параметрам.

**2.3. Требования к организации выходных данных**

Выходные данные сервера являются ответами на https запросы кассового и административного модуля и передаются в формате JSON согласно формату их входных данных.

**3. Приложение для администрирования**

1. **Требования к составу выполняемых функций**
   1. Возможность создания скрипта на языке JavaScript для вывода на кассу требуемой информации в зависимости от поступающих данных. Формат запросов содержится в разделе “Форматы”
   2. Возможность вывода информации о количестве людей, распознанных за определённый промежуток времени
   3. Возможность определения количества новых/существующих пользователей за определённый промежуток времени
   4. Построение графиков для значений, обозначенных в пунктах 1.2 и 1.3
2. **Требования к входным параметрам**
   1. Ответ на запрос о текущем состоянии обработчика получения информации. Формат запроса описан в разделе “Форматы”
   2. Ответ на запрос о событиях распознавания. Формат ответа JSON:

* массив, состоящий из элементов, описывающих произошедшие события
  + *date:* дата и время распознавания
  + *id:* уникальный идентификатор покупателя

1. **Требования к выходным параметрам**
   1. Запрос на изменение состояния обработчика получения информации. Формат запроса описан в разделе “Форматы”

Форматы:

1. Структура JSON строки при получении/отправке информации о текущем обработчике входных данных:

* *script\_text:* текст скрипта

Функции, указанной в тексте *JavaScript*, передаются следующие параметры

* *visit\_count:* количество посещений места клиентом

Функция возвращает текст, выводимый на экран оператору кассового аппарата

****