SIDEBAR STALKER

Приложение для проверки наличия студентов с помощью удостоверения, улучшенное с помощью модуля подсчета студентов.

Sebastian Niespodziany
Michał Sibera
Bartłomiej Wierzbiński
Dawid Wojciechowski

Содержание

О проекте

Темой проекта было выбрано приложение, созданное для проверки наличия студента на основании студенческого билета, однако, это не был наш первый выбор.

Изначально предложенный проект не находился среди возможных до реализации проектов. В основе лежала идея создания системы локализации внутри здания с помощью сигнала bluetooth, излучаемого с помощью маячков. Однако из-за дороговизны оборудования, а также ограниченного и неудовлетворяющего контента, проект была приостановлен. В конечном итоге было принято решение расширить тему. Мотивацией этого действия был тот факт, что выполнение проекта в той же тематики, что и другие коллеги, было мало привлекательным. Желая выделиться, мы добавили к возможностям нашего проекта дополнительную функцию, позволяющую подсчитать количество студентов в зале с помощью IP-камеры. Это стало хорошей возможностью чтобы напомнить себе навыки, полученные во время четвертого семестра, в ходе которого у нас проходили занятия на тему методов обработки изображений в языке python.

Но всё-таки после долгих дискуссий, языком для выполнения основного приложения был выбран javascript, в которым большинство нашей команды хотела-бы совершенствоваться. Выбирая другие темы у нас не было-бы такой широкой гаммы возможности.

Распределение работ

Реализатор	задача
Sebastian Niespodziany	● Модуль обнаружения и подсчета людей с помощью IP-камеры
Michał Sibera	● Основное приложение ● Перевод интерфейса
Bartłomiej Wierzbiński	● Модуль считывания данных со студенческих удостоверений
Dawid Wojciechowski	● Основное приложение ● Экспорты ● Связь между процессами

Используемые технологии

Node

Выбран в основном из-за менеджера пакетов. Остановившись на node, мы были почти

уверены, что найдем библиотеки, облегчающие реализацию проекта.

Npm

Система пакетов для node позволяющая в очень простой способ скачать и управлять

готовыми пакетами для решения проблемы без необходимости

писать все с самого начала.

Git

Популярная система контроля версий, позволяющая на параллельную работу с несколькими задачами одновременно. Дает возможность отмены ошибочно внесенных изменений или просто осуществить откат к предыдущей версии проекта.

Electron

Открытый Фреймворк для создания собственных кроссплатформенных приложений с использованием языков HTML, CSS, NodeJS. Кроме того, имеет большое сообщество сторонников, благодаря чему легко найти решения проблем при разработке программного обеспечения.

VueJS

Фронтэндовый Фреймворк, соединяющий все лучшее из фреймворков ReactJS и AngularJS. Сравнительно простой в освоении, при этом очень эффективный в

написании приложений. Имеет большое количество библиотек и плагинов, доступных в репозитории NPM (Node Package Manager).

Python 3.4 + OpenCV

Язык и библиотека, позволяющие на обработку изображений. Выбран из-за факта изучения его во время 4 семестра обучения. Использование его в проекте было хорошей возможностью для развития своих навыков в обращении с этими инструментами.

Webpack

Это так называемый bundler - инструмент для чтения всех файлов javascript и

Управляющий ими согласно определенным действиям в файле конфигурации. В нашем случае Webpack используется в первую очередь для:

● Соединения между собой сценариев в один, чтобы сэкономить на задержках

вытекающих из постановки нескольких небольших файлов, которые связаны с

отдельными запросами HTTP к серверу.

● Транспилации кода javascript с помощью инструмента Babel для его

новейших стандартов (например, ES6)

● Транспилация SaSS (Syntactically Awesome Style Sheets) для CSS

Выбирая технологию, мы руководствовались в первую очередь желанием попробовать новых возможностей создания приложений типа Desktop. Как базу проекта мы выбрали Electron из-за его растущей популярности в сфере создания приложений.

Выбирая Electron, нам пришлось быстро проанализировать есть-ли в NodeJS библиотеки для чтения электронных карт. Оказалось, что хранилище NPM (Node Package Manager) содержит довольно большое количество данных библиотек - поэтому именно с помощью electron мы решили общаться со считывателем благодаря ивентом javascript.

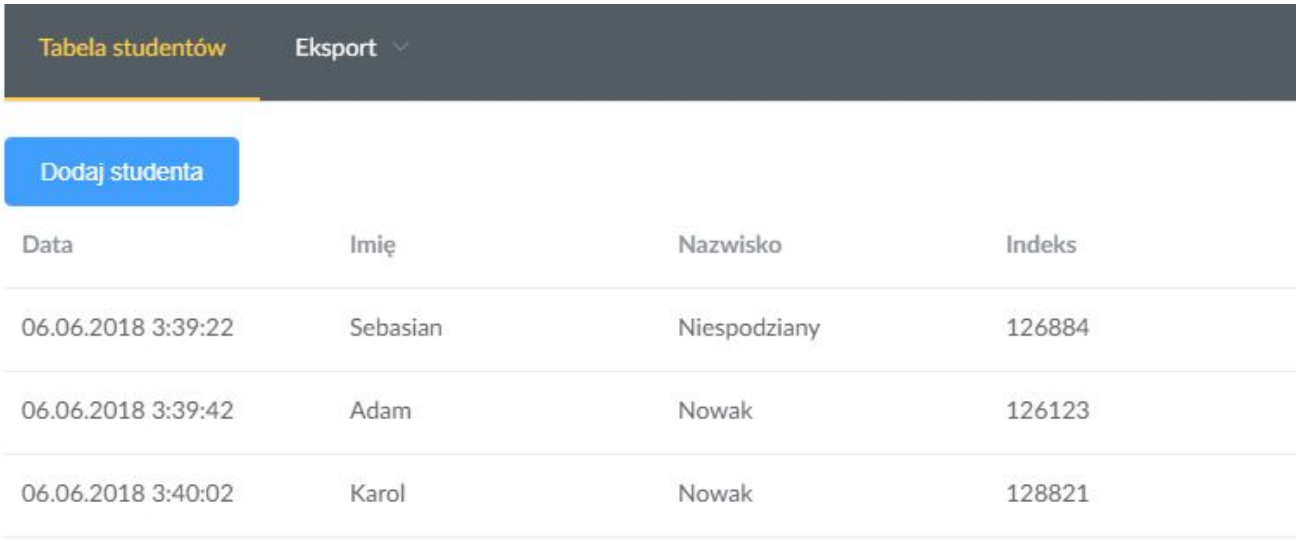
После выбора Electron, нам пришлось подобрать соответствующий Фреймворк и библиотеки, которую мы собираемся использовать для фронт-энда. Мы не хотели пойти по легкому пути воспользовавшись устаревшему уже в наше время JQuery и приложив к нему framework Bootstrap, мы начали задумываться над Фреймворком javascript. Выбор пал на VueJS из-за простоты создания приложений и сочетание всего, что есть хорошего с ReactJS и AngularJS. Для стилизации приложения мы решили использовать SaSS, чтобы сэкономить на повторении строк кода CSS. Кроме того, мы решили использовать ElementUI как библиотеку с базовыми стилями для тегов HTML. Всё это значительно облегчило создание последовательного интерфейса. ElementUI является решением аналогичной bootstrapa, предлагающего базовую конфигурацию стилей тегов HTML и много других услуг, как, например, валидация форм.

В начале мы планировали создать приложение в такой конфигурации желая использовать TypeScript – очень гибкий язык, который является транспилованым к Javascript. Быстро выяснилось, что существуют typings (определения классов Typescriptowych для Electrona и Vue. Даже нам удалось найти соответствующий Boilerplate для нашего приложения, использующего все, что нам было необходимо для создания нашего приложения. Однако появилась проблема с определениями для Vue. Мы нигде не могли найти несколько хороших советов, как использовать классы, чтобы настроить соответствующим образом, проект (мы хотели выделить в отдельных файлах отдельные конфигурации, составляющие основу проекта -, т. е. Vuex, Vue-маршрутизатор и т. д.). Попытка настройки проекта самостоятельно, положившись на удачу и угадав во многих местах, как следует хранить и экспортировать необходимые составные элементы классов (например, определение конфигурации Vuex имела в отдельных файлах показания, акции и мутации, которые были экспортированы в отдельный файл конфигурации, который только что был загружен в файл с базовой конфигурацией Vue) закончилась массой ошибок и конфликтов, которые привели к отказу от этой идеи и написанию кода непосредственно в JavaScript.

После выбора всех технологий, мы решили найти какой-то базовый проект в vue-cli (инструмент для инициализации проектов, основанных на VueJS). Здесь мы поставили на шаблон доступный на github, который создал пользователь SimulatedGreg. Этот шаблон содержал все, что нам было нужно для настройки в Webpacku (который является инструментом довольно трудно в освоении из-за большой трудности и сложности - инструмент предлагает очень большой спектр услуг для оптимизации проектов).

Функциональность приложения

Определение присутствия



Основной функционал приложения, представляющий студентов, присутствующих на занятиях была реализована в виде таблицы. Показаны в ней следующие информация:

- Дата и время подтверждения присутствия,
- имя студента,
- фамилия учащегося,
- индекс студента.

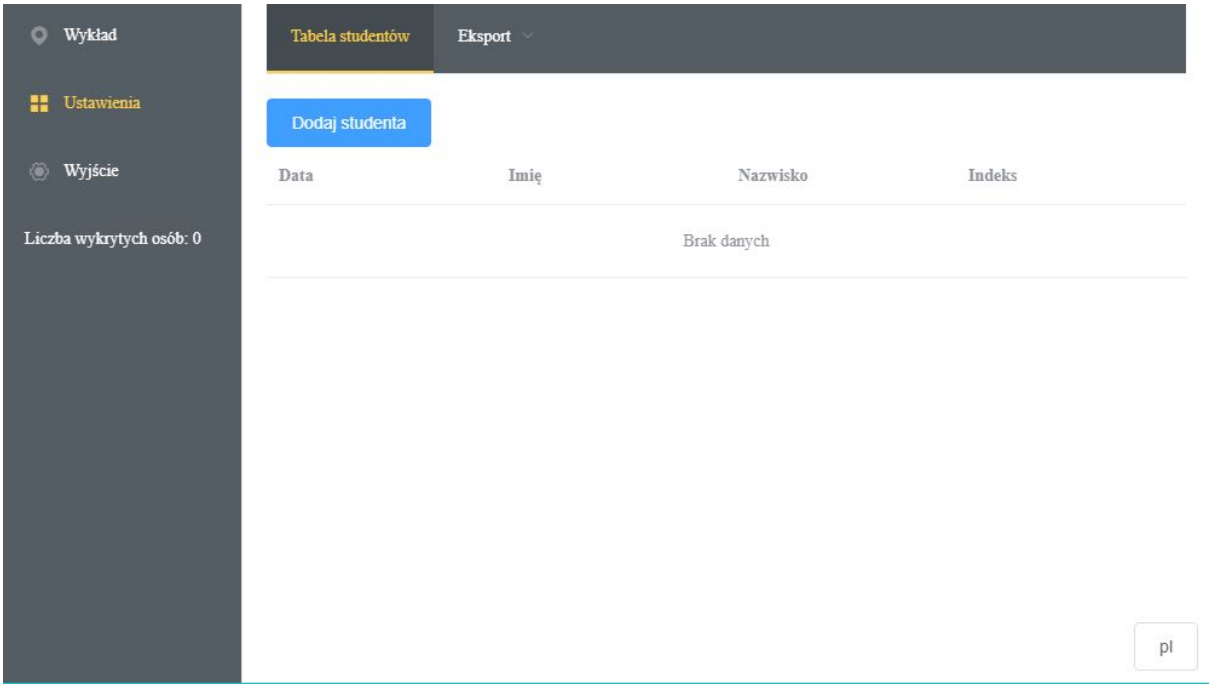
Дополнительные функции приложения расположены на боковой панели слева стороны, на верхней панели и в главном окне приложения. Это: кнопка подсчета нынешних людей через IP камеру, панель параметров, панель экспорта списка присутствия в файл, кнопку ручного добавления студента и изменение языка.

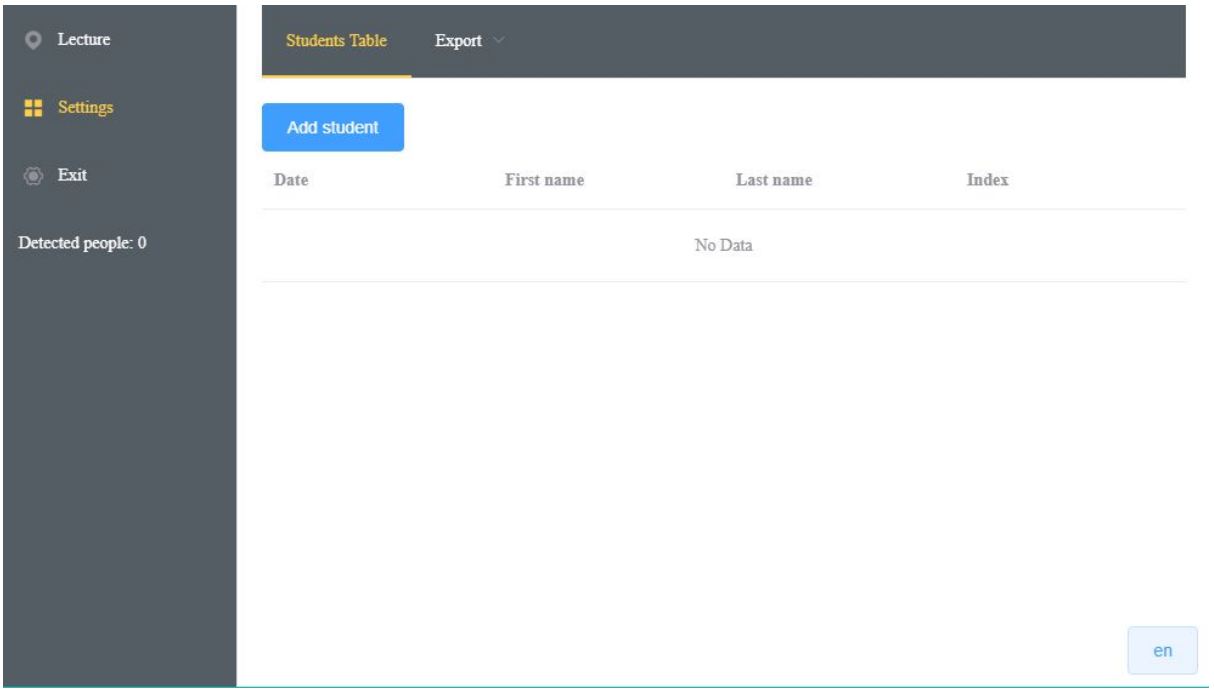
Все функции будут описаны по порядку, в следующей части документации.

Экспорт данных

Для архивирования списков присутствия была добавлена возможность экспорта списка в форматы CSV и XLS. Это популярные форматы, которые часто используются в примере как ведущие в процессе хранения и обработки данных. Их можно импортировать в такие программы, как, например, Microsoft Excel, в котором можно, например, вести статистику студентов или ввод комментариев.

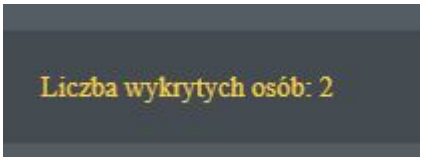
Другие языки





Дополнительная функция, позволяющая отображать интерфейс приложения на двух языках: польском и английском.

Подсчет людей

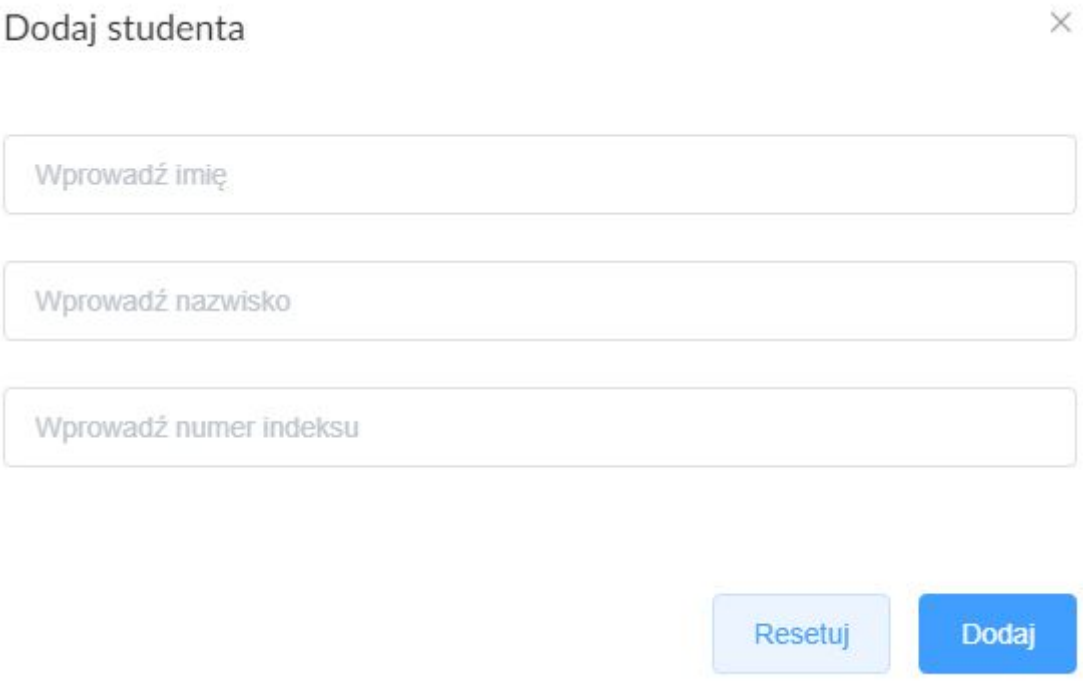


Функция, позволяющая совершить проверку количества людей, находящихся в зале. Система подсчитывает объекты, принадлежащие к классу человек. В случае большего количества человек, или узких залов результат следует рассматривать как относительный, нежели фактический показатель количества человек.

Параметры

Настройка соединения с камерой была упрощена до двух параметров. Чтобы подключиться к камере необходимо ввести IP-адрес и используемый порт.

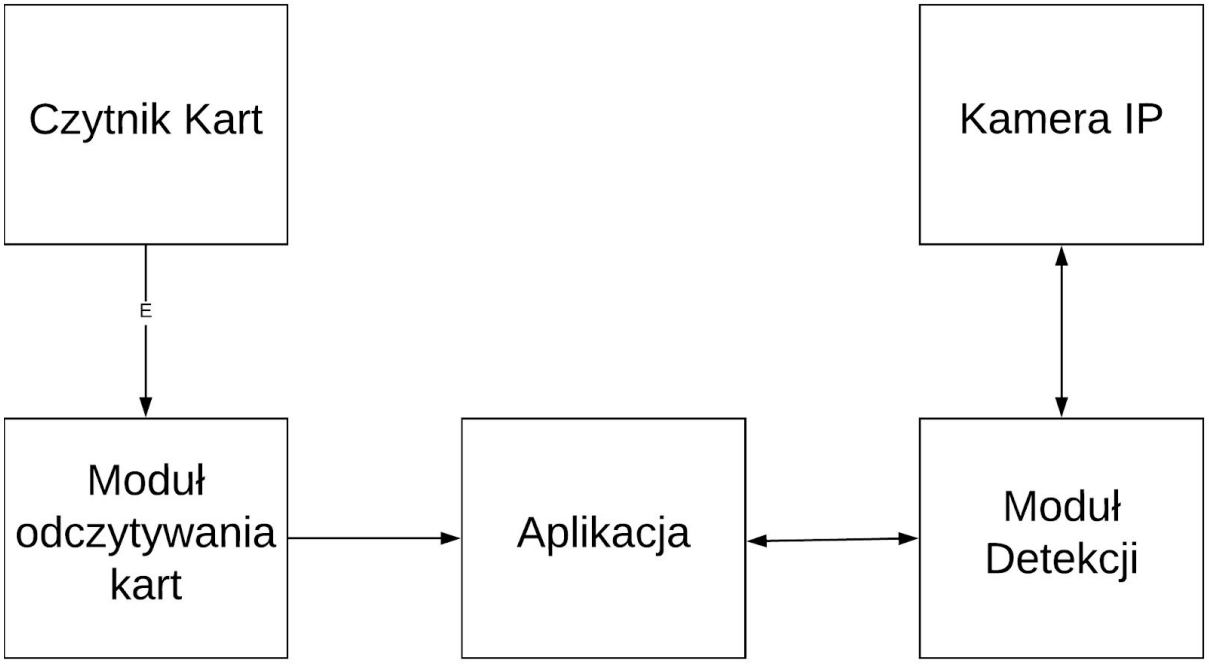
Форма для ручного добавления студента



Дополнительная функция, позволяющая добавить студента без необходимости использования студенческого билета. Это безопасный способ добавления присутствии в случаях, как, например: повреждённое удостоверение, неподдерживаемый формат карты, или её отсутствия. Добавление студента в список осуществляется с помощью встроенной формы.

Архитектура решения

Схема работы приложения

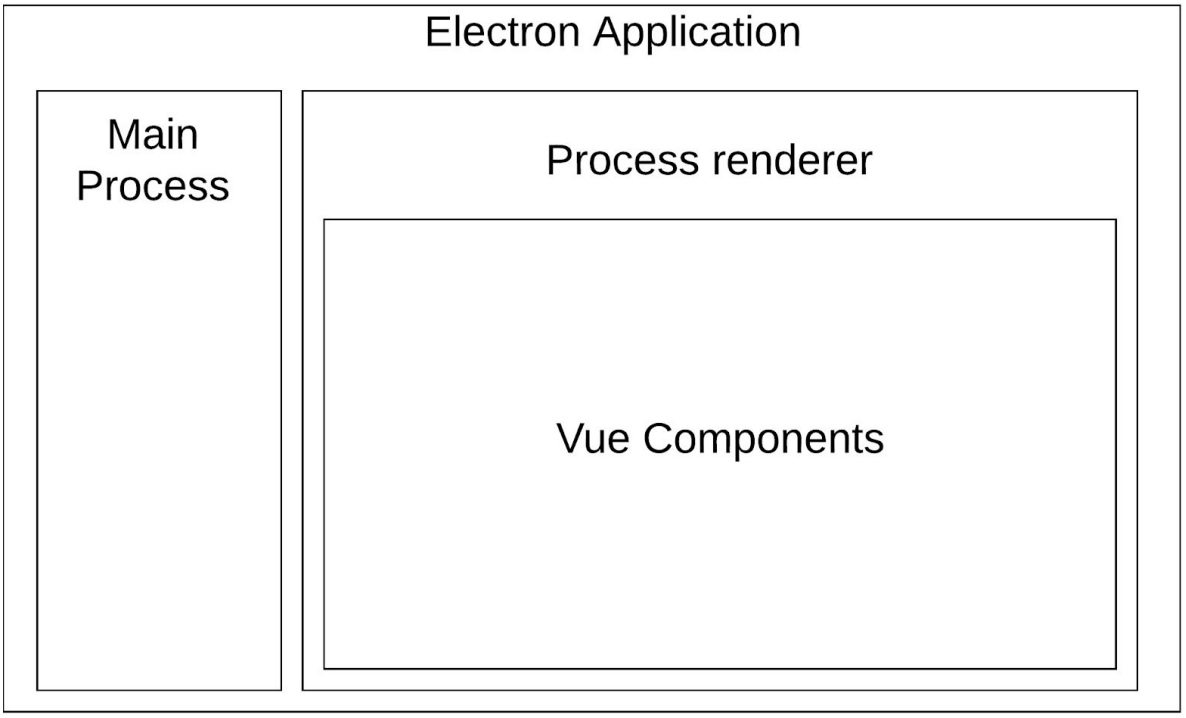


Датчик регистрации карт – модуль считывателя карт – приложение – модуль обнаружения – камера IP

Модуль чтения карт, получает данные, хранящиеся на карте и при помощи отправки соответствующей команды, передает данные в приложение.

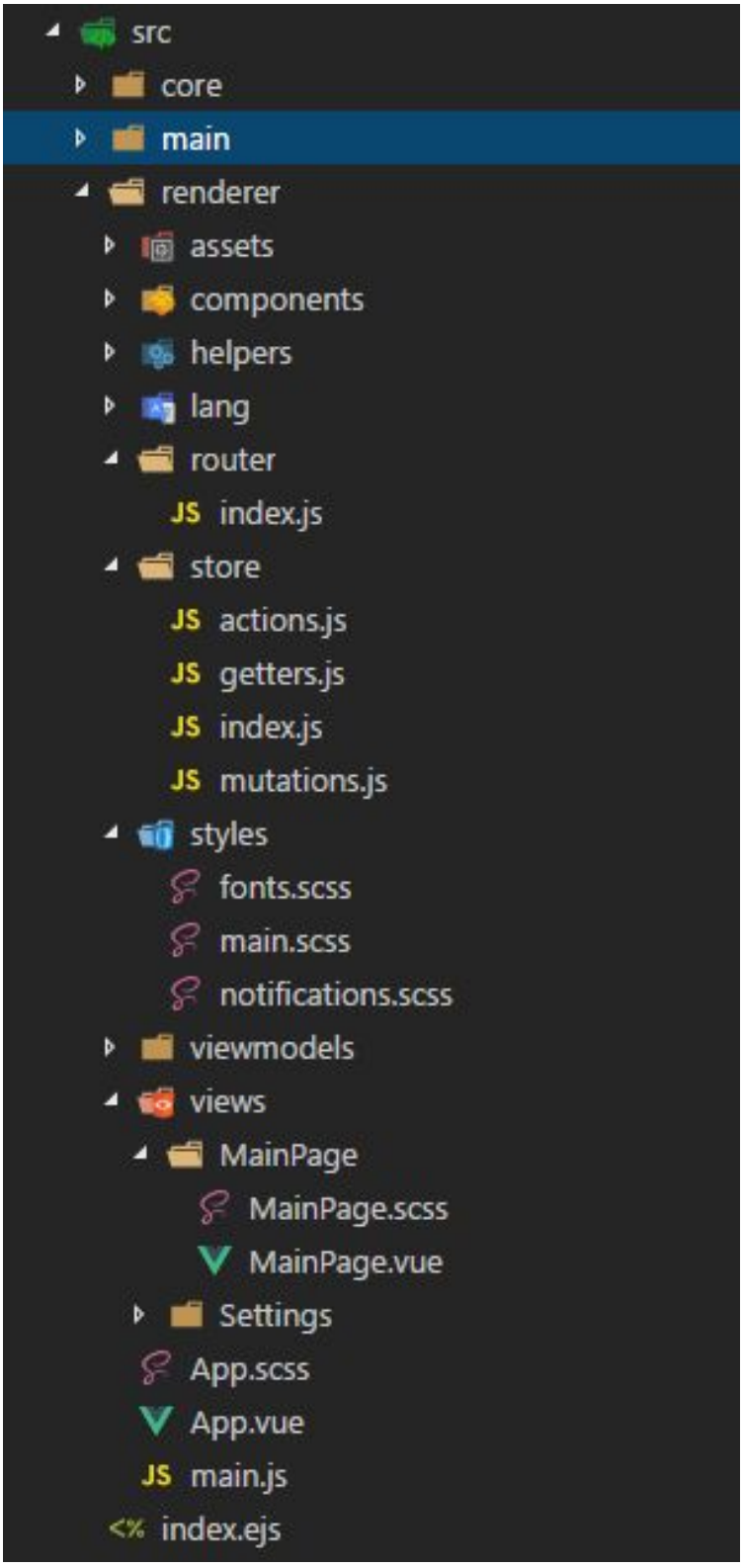
Модуль регистрации отвечает за соединение с IP-камерой, обработкой полученного изображения и вычисление количества студентов, которые находятся в зале.

Модуль приложение отвечает за обработку собственных входящих мероприятий и 2 остальных модулей, а также отображение интерфейса пользователя. Для того, чтобы избежать эффекта так называемой "заморозки интерфейса" приложение использует 2 процессы.



Основной процесс, отвечает за обслуживание мероприятий, расчета, изменение данных и т. д. А задачами процесса визуализации является визуальная поддержка GUI и просмотр текущих данных, содержащихся в Vue Storage. Информация отображаются пользователю приходят с плагином i18n для VueJS для определения структур, позволяющих создавать изображение независимо от языка. Для добавления новых языков достаточно создать новую структуру интеллектуального ввода текста для конкретного языка, и добавить в настройки. Кроме того, приложение использует так называемый Vuex Store для хранения данных и обмена между компонентами. Все изменения состояния приложения происходят в соответствии с огульно принятой конвенцией. Сначала вызывается из вида, действие (которое отвечает за структуру данных, например, скачивание и обработку). Затем действие вызывает мутацию (которая отвечает за изменения состояния). Ссылки на состояния проходят через getters (также определены в store).

Структура каталогов проекта



В исходной папке приложения находятся следующие каталоги:

● core - содержит определения базовой функциональности приложения, т. е. файлы исходных сценариев обработки изображения и карт-ридера.

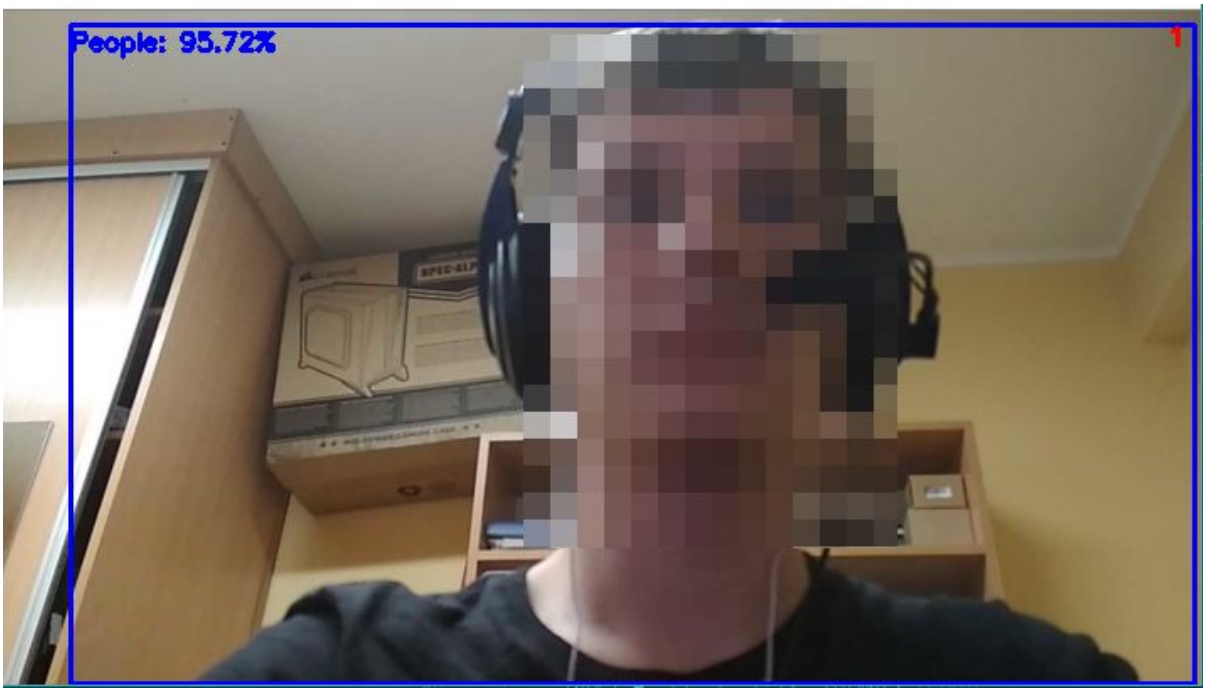
● main - содержит определения основных сценариев запуска Electrona.

● renderer - содержит определения процесса визуализации представления.

- ○ assets - включает в себя статические элементы, такие как изображения и шрифты
- ○ компоненты - содержит компоненты vue, которые используются
- ○ как отдельные элементы многократного использования (например, заголовок в интерфейсе, боковое меню).
- ○ helpers - включает в себя сценарии, которые импортируются в
- ○ компоненты vue и в store.
- ○ lang - содержит определение двигателя i18n наряду с определением конкретных
- ○ языков, в отдельных файлах, содержащих структуру, типа json (например, en.js)
- ○ маршрутизатор - содержит конфигурацию путей маршрутизации приложения
- ○ store - содержит конфигурацию Vuex Store. В отдельных файлах были созданы, getters, акции и мутации, которые импортируются в файл для - index.js
- ○ styles - включает в себя настройку стилей основного приложения (шрифт,
- ○ уведомления и т. д.).
- ○ viewmodels - содержит вспомогательные классы, являющиеся моделями вида приложения.
- ○ views - содержит компоненты vue, которые используются в качестве основных видов, в которые устанавливаем компоненты из папки “components”.

Определение числа студентов

Дополнительная функция, позволяющая определить количество людей, находящихся в поле зрения IP-камеры. Расчет осуществляется путем передачи извлеченных контуров объектов на вход модели и подсчет экземпляров объектов, отнесенных к классу человек. Использованным модель MobilNetSSD, использующий метод Single Shot Detection. Эта модель классификация более 30 объектов, в том числе и людей.



Из-за большой нагрузки процессора во время операций классификации, возможность обнаружения людей была реализована как сервис по запросу. При нажатии на обнаруженную количество людей вызывает соединение с камерой на время загрузки одного кадра и проведение операций подсчета, студентов как раз захваченного кадра.

Считывание данных со студенческих карт

Модуль считывания данных с удостоверения, был написан с помощью electrona и используя следующие библиотеки NPM.

- Smart-card - API для отправки и получения команд APDU
- asn1js - parser объектов ASN.1

Электронный студенческий билет совместим со стандартом ISO 7816, а данные хранятся в формате ASN.1 БЕР.

Данные на карте организованы в древовидную структуру. Там находятся три типа файлов:

- MF (Master File) - корень дерева
- DF (Directory File) - аналог каталога
- EF (Elementary File) - обычный файл

Каждый из этих файлов имеет двухбайтовый адрес.

Сканирование студенческого билета раскрывает перед нами следующие файлы:

- 0x0101 (DF.SELS) - Электронный Студенческий Билет
  - 0x0001 TRANSPARENT (EF.CERT) - квалифицированный подпись эмитента
  - 0x0002 TRANSPARENT (EF.ELS) - файл с данными удостоверения + подпись

Чтение данных из файлов осуществляется через интерфейс APDU. Команда APDU имеет следующую структуру:

Команда APDU						
CLA	INS	P1	P2	Lc	DATA	Le

Поле	Имя	Длина	Описание
CLA	Class	1 байт	Класс инструкции
INS	Instruction	1 байт	Конкретная инструкция для исполнения
P1	Parameter 1	1 байт	Первый параметр инструкции
P2	Parameter 2	1 байт	Второй параметр инструкции
Lc	Length command	0 - 3 байта	Количество байтов с данными
Data	Data	Lc байтов	Данные
Le	Length expected	0 - 3 байта	Макс. количество байт предполагаемые в ответе

Ответ APDU имеет следующую структуру:

Odpowiedź APDU		
Data	SW1	SW2

Поле	Имя	Длина	Описание
Data	Data	Макс. Le байтов	Данные возвращены через карту
SW1	Status word 1	1 байт	Код статуса ответа
SW2	status word 2	1 байт	Код статуса ответа

Получение данных из Электронного Студенческого билета-это процесс многоступенчатый. Первым шагом является навигация в дереве файлов в соответствующего объекта DF. Для этого используется функция selectFile библиотеки smartcard, которая принимает в качестве параметра массив байтов. В случае студенческого билета это [0xD6, 0x16, 0x00, 0x00, 0x30, 0x01, 0x01].

Выполняется команда APDU со следующими параметрами:

Поле	Величина(hex)
CLA	00 (NO SECURE MESSAGE)
INS	A4 (SELECT FILE)
P1	04
P2	00
Data	D6 16 00 00 30 01 01

Затем выбирается соответствующий файл EF. В случае ELS, данные о студенты находятся в 0x0002 TRANSPARENT (EF.ELS). Команда APDU имеет таким образом следующий вид: [0x00, 0xA4, 0x02, 0x00, 0x02, 0x00, 0x02, 0x12]

Поле	Величина(hex)
CLA	00
INS	A4 (SELECT FILE)
P1	02
P2	00
Lc	02
Data	00 02
Le	12

На последнем шаге необходимо прочитать содержимое файла 0x0002 TRANSPARENT (EF.ELS). Нет необходимости чтения данного файла. Конкретно интересует нас фрагмент, в котором находятся данные о студенте. Благодаря этому, чтение происходит быстрее, однако, вы должны принять дополнительные меры во время обработки данных.
Команда APDU имеет вид [0x00, 0xB0, 0x00, 0x00, 0xF8].

Поле	Величина(hex)
CLA	00
INS	B0 (READ BINARY)
P1	00
P2	00
Data	F8

После получения ответа следует извлечь объект ASN.1 с данными студента. Фрагмент с кодом 2a8468016504010101 является его началом. Конец фрагмента следует рассчитать на основании данных, содержащихся в последовательных байтах в соответствии со структурой ASN.1.

При помощи parsera (библиотеки) asn1js следует выбрать поля с именем, именем и индексом, и передать их в основную программу для просмотра.

Интересные задачи

Отсутствие документации студенческого билета

Написав модуль чтения студенческого билета, напрасно искать документации этой системы. Необходимые команды APDU, список файлов или описания применяемых
схем взяты с форумов и блогов людей, которые работали
с подобной задачей. Полезной была также приложение SmartCardSuite, в которой
журналах можно было увидеть передаются команды APDU.

Проблемы, связанные с обновлением Windows 10

В процессе работы над проектом, Microsoft выпустил новое обновление для системы
Windows 10 (версия 1803). Обновление системы до последней версии
вызывало проблемы с используемой библиотекой для обслуживания устройств чтения карт памяти.
Для наиболее часто встречающихся ошибок можно отнести, невозможность отправки
команды APDU, отсутствие ответа на команду или-же остановка всего
процесса чтения данных.

Инструкция по эксплуатации

Настройка камеры

Желая воспользоваться функцией подсчета людей, вам необходимо заполнить параметры
настройки в разделе настройки приложения. К ним относятся порт и IP-адрес камеры. После сохранения настроек, функция становится активна.

Приложение

Приложение после запуска сразу готов к работе. После подключении
ридера можно приступить к сканированию карт. Последующие сканы будут появляться
в главном окне приложения вместе с визуальным уведомлением правильности
сканирования в виде pop-up'а.

Используемая литература

- Документация Vue.JS - https://vuejs.org/v2/guide/index.html
- Документация Webpack - https://webpack.js.org/concepts/
- Документация OpenCV - https://docs.opencv.org/3.0-beta/index.html
- Npm - https://www.npmjs.com
- считывание студенческого электронного билета - http://www.makdaam.eu/2008/04/czytanie-els-przez-interfejs-stykowy/