# Особенности Electron

В связи с тем, что Electron работает на основе Chromium, последние версии фреймворка больше не поддерживают ОС ниже, чем WINDOWS 10.

В зависимости от нужд проекта, можно собрать приложение для Windows, Linux или MAC OS.

# Пути проекта

**/html** - хранятся все html страницы приложения.

**/css** - хранятся все css страницы приложения.

**/js** - хранятся все js файлы приложения.

/img - хранятся все необходимые изображения.

## HTML

На данный момент создано 6 страниц на каждую задачу по работе с данными.

### Распределение нагрузки

Основная страница, на которой пользователь распределяет нагрузку между преподавателями по году, форме обучения, семестру.

### Статистика о преподавателях

Предпросмотр статистики о преподавателях по определённым фильтрам. В будущем планируется добавить возможность предпросмотра и создания статистики в формате *docx* файла. Связано с распределением нагрузки.

### Учебный план

Добавление и изменение предметов, связано с потоками, дисциплинами и типами.

### Дисциплины

Добавление и изменение дисциплин.

### Типы

Добавление и изменение типов предметов (пример: лекции, практики и т.д.).

### Потоки и Группы

Добавление и изменение потоков и групп учеников, к потоку прикрепляются указанные группы.

### Преподаватели кафедры

Добавление и изменение преподавателей, в последствии используется в распределении нагрузки.

## CSS

Файлы css для настройки стилей html страниц.

**style.css** - главный css файл с основными настройками стилей, импортирует в себя все второстепенные.

**style-fonts.css** - для импорта пользовательских шрифтов.

**style-table.css** - стили для таблиц.

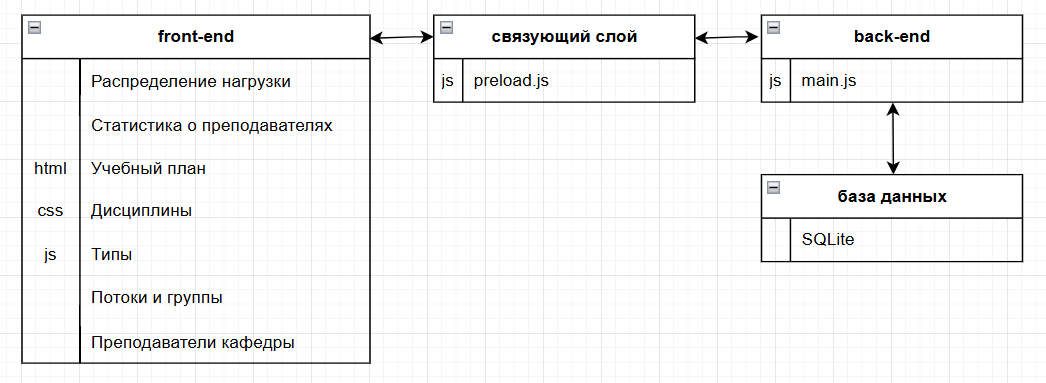
**style-sidebar.css** - стили для бокового меню.

## JavaScript

Файлы JavaScript на фреймворке можно разделить на три ключевых типа:

1. **main.js** - главный скрипт, выполняющий роль backend сервера, через него подключается база данных и выполняется главная логика приложения.
2. **render\*.js** - скрипт для страниц HTML, предназначен для работы приложения со стороны клиента. (\* - для каждой страницы создаётся свой render.js, вместо \* подставляется слово-обозначение для отличия от остальных файлов).
3. **preload.js** - скрипт прослойка между render.js и main.js, выполняет роль моста, который связывает пользователя с локальным сервером и передаёт данные в виде json данных.

Схема взаимодействия всех файлов:



### main.js

Подключённые модули nodejs:

1. sqlite3 - библиотека для работы с базой данных SQLite.
2. xlsx - библиотека для работы с xlsx файлами.
3. fs - файловый менеджер для работы с файлами.

**app.whenReady()** - метод, запускающийся при старте приложения.

**createWindow()** - метод, создающий окно приложения.

- BrowserWindow - объект, внутри которого указан словарь с параметрами для создаваемого окна:

width - ширина,

height - высота,

webPreferences - особенности веба

preload - путь до preload.js

BrowserWindow.loadFile(path) - метод для загрузки страницы. В текущем положении запускает index.html, так как метод вызывается при запуске приложения.

path - путь до запускаемого html файла.

**ipcMain.handle(name, params)** - метод, принимающий данные из preload.js.

- name - название пути, который принимает метод.

- params - параметры, которые получает метод.

**checkDatabaseTables()** - метод, проверяющий существование всех необходимых таблиц.

**getX, setX, updateX, deleteX** - методы для добавления, изменения, удаления, обновления данных в таблицах.

preload.js

Предоставляет безопасный API для взаимодействия между renderer-процессом и main-процессом.

*Методы:*

1. Для получения данных (get...)
2. Для обновления данных (update...)
3. Для вставки данных (insert...)
4. Для удаления данных (delete...)

### renderDisciplines.js

Управление таблицей дисциплин с возможностью добавления, редактирования и удаления записей.

*Основные функции:*

1. getDataAndCreateTable() - получает данные о дисциплинах и создает таблицу
2. createTableOfDisciplines(data) - рендерит таблицу дисциплин
3. updateCurTable() - обновляет таблицу

*Элементы управления*:

1. Кнопка добавления новой дисциплины
2. Кнопки редактирования/сохранения для каждой строки
3. Кнопки удаления для каждой строки

*IPC-взаимодействие:*

1. getDisciplines() - получение списка дисциплин
2. insertDiscipline(data) - добавление новой дисциплины
3. updateDisciplinesTable(data) - обновление дисциплины
4. deleteDiscipline(data) - удаление дисциплины

renderFlows.js

Управление потоками и группами студентов.

*Основные функции:*

1. updateTables() - обновляет таблицы потоков и групп
2. createTableFromDatabase(database, containerID) - создает таблицу потоков или групп

*Элементы управления:*

1. Кнопки добавления/удаления потоков и групп
2. Редактируемые ячейки таблицы

*IPC-взаимодействие:*

1. getDatabaseTable('flows') - получение списка потоков
2. getGroups('') - получение списка групп
3. insertFlow(data) - добавление потока
4. insertGroup(data) - добавление группы
5. deleteFlow(data) - удаление потока
6. deleteGroup(data) - удаление группы

renderKafedra.js

Управление кафедрой и преподавателями.

*Основные функции:*

1. updateCurTables() - обновляет список преподавателей
2. updateTablePersonalInfo(data) - показывает детальную информацию о преподавателе
3. updatePersonalInfo(id) - обновляет информацию о преподавателе

*Элементы управления:*

1. Список преподавателей слева
2. Форма редактирования справа
3. Кнопки добавления/удаления/сохранения

*IPC-взаимодействие:*

1. getDatabaseTable('kafedra') - получение списка преподавателей
2. insertPerson(data) - добавление преподавателя
3. updateKafTable(data) - обновление данных преподавателя
4. deletePerson(data) - удаление преподавателя

renderPersonalHours.js

Просмотр персональной нагрузки преподавателей.

*Основные функции:*

1. updateSelectorTeachers() - обновляет список преподавателей
2. updateTeacherList(data) - показывает нагрузку выбранного преподавателя
3. createHeadRow() - создает заголовок таблицы

*Элементы управления:*

1. Выпадающие списки для выбора года, формы обучения и преподавателя
2. Кнопка получения данных

*IPC-взаимодействие:*

1. getCurTeachers([]) - получение списка преподавателей
2. getCurListTeachers(data) - получение нагрузки преподавателя

renderSyllabus.js

Управление учебными планами (syllabus).

*Основные* функции:

1. createTableFromDatabase(database) - создает таблицу учебных планов
2. updateCurTable() - обновляет таблицу

*Элементы управления:*

1. Кнопки добавления/редактирования/удаления
2. Форма с выпадающими списками для потоков, дисциплин и типов занятий

*IPC-взаимодействие:*

1. getSyllabus([]) - получение учебных планов
2. getFlows(dataTo) - получение потоков
3. getDisciplines(dataTo) - получение дисциплин
4. getTypes(dataTo) - получение типов занятий
5. insertSyllabus(data) - добавление учебного плана
6. updateSyllabusTable(data) - обновление учебного плана
7. deleteSyllabus(data) - удаление учебного плана

renderTypes.js

Управление типами занятий.

*Основные функции:*

1. getDataAndCreateTable() - получает данные и создает таблицу
2. createTableOfDisciplines(data) - рендерит таблицу типов
3. updateCurTable() - обновляет таблицу

*Элементы управления:*

1. Кнопки добавления/редактирования/удаления

*IPC-взаимодействие:*

1. getTypes() - получение типов
2. insertType(data) - добавление типа
3. updateTypesTable(data) - обновление типа
4. deleteType(data) - удаление типа

# База данных:

Используется SQLite.

Основные таблицы: kafedra, flows, groups, disciplines, types, syllabus, personal\_plan