**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

Рассмотрим подробно функционирование программы. Для этого проведем анализ основных модулей программы и рассмотрим их зависимости. А также проанализируем все модули, которые входят в состав кода программы, и рассмотрим назначение всех методов и переменных этих модулей.

В разрабатываемом веб-приложении можно выделить следующие модули:

* ;

Изначально пользователь попадает на страницу авторизации логика которой определена классом LoginPage, при помощи этого класса пользователь может авторизоваться и приступить к работе с системой. Далее пользователь попадает на главную страницу системы где всю логику реализует класс Dashboard. Навигация по приложению происходит при помощи класса Toolbar. При переходе на страницу листинга студентов логика поведения приложения задается классами: StudentsList, StudentsListItem, StudentsEditorPage. Данные классы используются для отображения и редактирования студентов. На странице преподавателей в свою очередь используются классы: TutorsList, TutorsListItem, TutorsEditorPage. Выше приведенные классы запрашивают данные студентов и преподавателей и отображают их на соответствующих страницах приложения. Так же они содержат в себе методы для редактирования и обновления данных на странице.

**3.1 Классы серверного модуля авторизации**

Для выполнения серверной части дипломного проекта была выбрана концепция промежуточного программного обеспечения. Основной единицей данного подхода является middleware. Ниже приведено их описание.

Инициализация промежуточного програмного обеспечения аутентификации и авторизации происходит при помощи метода initialize(), данный метод возвращает исходный обьект middleware который впоследствии мы будем конфигурировать в зависимости от наших требований безопасности и правил доступа.

Конфигурация уровня доступа к определенным частям приложения осуществляется методом unless() объекта jwt , в который необходимо передать пути которые будут доступны не авторизоанным пользователям, так же при инициализации объекта jwt необходимо определить приватную часть ключа для генерации валидных токенов, которые будут использоваться при аутентификации пользователей.

Для определения способа авторизации необходимо передать функцию которая будет производить проверку связки пароля и электронной почты пользователя, для этого используется метод authenticate(), ранее инициализированного объекта passport данный метод будет вызван при каждой попытке авторизации. Данный метод возвращает валидный токен который будет использоваться при каждом вызове API.

При попытке не авторизованного доступа необходимо сообщить об этом пользователю, для этих целей используется специальное промежуточное програмное обеспечение которое обрабатывает эти запросы. Оно возвращает ошибку HTTP протокола и сообщение ошибки валидации. Так же на этом уровне определим обработчик ошибок который будет сообщать пользователю об ошибках случившихся при обработке его запроса. Для этих целей будем использовать метод use()объекта controller . Данный метод принимает функцию в аргументах которой находятся обьекты req и res которые необходимы для отправки сообщения пользователю.

**3.2 Классы серверного модуля работы со студентами**

Для обработки запросов по изменению, созданию, получению и удалению студентов используется несколько классов: controller, model, service, validator.

**3.2.1 Класс StudentController**

Это класс который используется для обработки HTTP-запросов. Он представляет собой набор методов которые вызываются в зависимости от URL на который происходит запрос. Он состоит из нескольких методов которые приведены ниже. Контроллеры состоят из действий, которые являются основными блоками, к которым может обращаться конечный пользователь и запрашивать исполнение того или иного функционала. В контроллере может быть одно или несколько действий.

Метод Update используется для обновления записи студента в базе данных. Данный метод получает данные из валидатора и обращается к базе данных для их записи. Так же данный метод получает id студента из параметров HTTP-запроса.

Метод FillDiplomaDays используется для заполнения расписания консультаций по дипломному проектированию для каждого студента. Данный метод получает на вход в качетсве параметра доступные даты, а так же временные промежутки для каждой даты для равномерного распределения посещений студентами консультанта. Данный метод находит порционное количество студентов на единицу времени, затем при помощи этих данных формируется функция распределения для каждого студента, где вероятность попадания в какой-либо день зависит от порционного количества студентов на единицу времени, чем больше это число, тем больше вероятность того что консультация для этого студента выпадет именно в этот день. Таким образом формируется равномерное распределение студентов по заданным дням консультаций. Далее полученные данные сохраняются в базу данных.

Метод FillPracticeDays используется для заполнения расписания консультаций по преддипломной практике для каждого студента. Данный метод использует тот же алгоритм что и FillDiplomaDays. Полученные данные так же сохраняются в базу данных.

Метод GetAllStudents используется для получения списка всех студентов. Данный метод вызывает метод List у объекта StudentService.

Метод UploadStudents принимает файл c расширением \*.xlsx, парсинг которого производится при помощи библиотеки XLSX, данная библеотека возвращает список студентов в формате JSON, которые в последствии будут переданы объекту StudentsService для дальнейшей записи в базу данных.

Метод GetStudentByID возвращает обьект студента полученный при помощи переданного ID.

**3.2.2 Класс StudentValidator**

Данный класс используется в качестве валидатора данных которые необходимо записать в базу данных. Данный класс содержит 2 метода ValidateUpdate и ValidateCreation. Каждый метод вызывается в зависимости от тип действия, при обновлении ValidateUpdate, а при создании в свою очередь ValidateCreation. Данный класс использует класс validation. Данный класс предоставляет набор методов для валидации строк и чисел, приведем несколько функций:

- isPhoneNumber () – позволяет определить, является ли переданная строка номером телефона.

- maxLength () – позволяет максимальную длинну строки.

- isEmail () – позволяет определить, является ли переданная строка адресом электронной почты.

**3.2.3 Класс StudentModel**

Данный класс представляет собой модель. В нем содержится поведение и методы необходимые для взаимодействия со студентами. Класс наследуется от Mongoose тем самым он сопоставлен с соответствующей таблицей в базе данных.

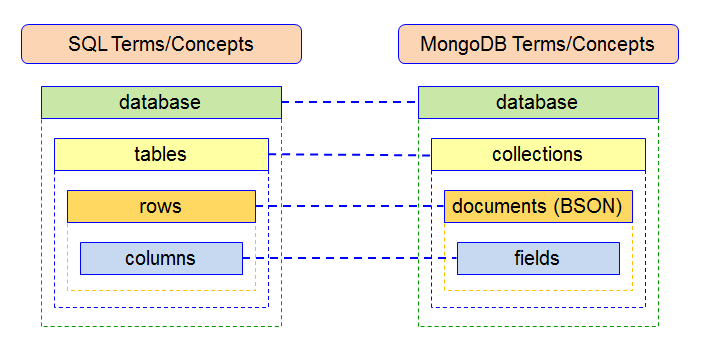


Рисунок 3.1 – Сравнение ODM и SQL модели

Приведем некоторые методы которые предоставляет данная модель:

- find() – позволяет найти записи при помощи запроса;

- map() – позволяет задать функцию которая будет применена ко всем записям в таблице;

- add() – добавить пару ключ-значение в модель;

- findById() – позволяет найти запись при помощи ID;

- insertMany() – позволяет записать массив данных в таблицу;

- create() – создает новую запись в таблице;

**3.2.4 Класс StudentService**

Данный класс является промежуточным звеном между контроллером и моделью, он инкапсулирует всю специфичную логику которая необходима для модели: мапинг параметров, обработка ошибок и подготовка данных для ответа пользователю. Сервис предоставляет следующий набор методов:

- getById() – возвращает запись студента с определенным ID;

- getAll() – возвращает список всех студентов;

- update() – возвращает обновленного студента;

- insert() – возвращает список обновленных студентов;

- create() – создает и возвращает созданного студента

**3.3 Классы серверного модуля работы с преподавателями**

Для обработки запросов по изменению, созданию, получению и удалению преподавателей используется несколько классов: controller, model, service, validator.

**3.3.1 Класс TutorsController**

Это класс который используется для обработки HTTP-запросов. Он представляет собой набор методов которые вызываются в зависимости от URL на который происходит запрос. Он состоит из нескольких методов которые приведены ниже.

Метод Update используется для обновления записи преподавателя в базе данных. Данный метод получает данные из валидатора и обращается к базе данных для их записи. Так же данный метод получает id преподавателя из параметров HTTP-запроса.

Метод GetAllTutors используется для получения списка всех студентов. Данный метод вызывает метод List у объекта TutorsService.

Метод UploadTutors принимает файл c расширением \*.xlsx, парсинг которого производится при помощи библиотеки XLSX, данная библеотека возвращает список студентов в формате JSON, которые в последствии будут переданы объекту TutorsService для дальнейшей записи в базу данных.

Метод GetTutorByID возвращает обьект преподавателя полученный при помощи переданного ID.