

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт информационных технологий и управления в технических системах
(полное название института)

кафедра «Информационные системы»
(полное название кафедры)

Пояснительная записка

к курсовой работе
по дисциплине «Технологии баз данных»

на тему «База данных электронный журнал старосты»

Выполнил: студент III курса, группы: ИС/б-31-о

Направление подготовки (специальности) 09.03.02

Информационные системы и технологии
(код и наименование направления подготовки (специальности))

профиль (специализация)

Куркчи Ариф Эрнестович
(фамилия, имя, отчество студента)

Руководитель Тимофеев И.С. ст. преподаватель

(фамилия, инициалы, степень, звания, должность)

Защита « » декабря 2016 г.

Оценка

Руководитель

(подпись)

(фамилия, инициалы)

Ведущий преподаватель

(подпись)

(фамилия, инициалы)

2016 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
ПОСТАНОВКА ЗАДАЧИ.....	4
1 АНАЛИТИЧЕСКАЯ ЧАСТЬ	5
1.1 Анализ предметной области	5
1.2 Анализ отношений между объектами.....	6
2 РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ БАЗЫ ДАННЫХ.....	8
2.1 Построение диаграммы «сущность-связь» в нотации П.Чена	8
2.2 Построение полной атрибутивной модели в нотации IDEF1X.....	9
3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ	11
3.1 Обоснование выбора СУБД.....	11
3.2 Разработка физической модели базы данных	11
3.3 Разработка запросов к базе данных	13
3.4 Разграничение прав доступа	14
3.5 Исследование информационных параметров базы данных	14
4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ.....	17
4.1 Обоснование выбора языка программирования	17
4.2 Разработка интерфейса пользователя	17
4.3 Тестирование работы приложения.....	26
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	28
ПРИЛОЖЕНИЕ А.....	29
ПРИЛОЖЕНИЕ Б	30

ВВЕДЕНИЕ

На сегодняшний день, очень затруднительно представить любую информационную систему, которая не использует базы данных. Их развитие было очень тесно связано с развитием автоматизированной обработки данных и увеличением количества различных категорий пользователей, что послужило к построению более мощных СУБД, имеющие средства защиты информации, высокую отказоустойчивость и производительность.

Целью данного курсового проекта является создание базы данных, используемой для упрощения исполнения обязанностей старостам. Для реализации данной работы, были проведены следующие этапы:

- анализ предметной области (определяются пользовательские системы, функциональные требования системы, выделяются основные сущности разрабатываемой системы и отношения между ними, на основе выделенных сущностей, определяются их атрибуты и строится ER-диаграмма);
- разработка логической схемы базы данных (построение и нормализация реляционной модели);
- реализация базы данных (обоснование выбора СУБД, разработка физической схемы базы данных, реализация и тестирование базы данных, запросов к ней, разграничение прав доступа);
- исследование информационных параметров базы данных (подсчитывание значений некоторых параметров БД и приложения в целом);
- разработка клиентского приложения (обоснование выбора языка программирования, разработка клиентского приложения, описание интерфейса пользователя, тестирование системы).

ПОСТАНОВКА ЗАДАЧИ

Для реализации базы данных электронного журнала старосты, а также клиентского приложения, для работы с разработанной БД, были поставлены следующие задачи:

- разработать логическую модель базы данных (построить диаграмму «сущность-связь» в нотации П.Чена, построить модели основанной на ключах, а также построить полную атрибутивную модель IDEF1X. Нормализовать полученную логическую модель до третьей нормальной формы);
- разработать физическую модель базы данных (выбрать аппаратную и программную платформу для реализации базы данных, а также её реализовать, протестировать полученную базу данных, разграничить права доступа, и рассчитать информационные параметры базы данных);
- разработать пользовательский интерфейс (выбрать и обосновать выбор языка программирования, разработать интерфейс пользователя, алгоритм работы каждого модуля, и протестировать разработанное приложение).

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ предметной области

Журнал старосты – это официальный документ, содержащий информацию об студентах группы, посещения ими занятий, общую статистику пропусков, а также информацию о занятиях. Староста занимается организацией студентов внутри группы и следит за посещаемостью занятий. Раз в 2 недели староста обязан подавать сведения о 5 студентах, чья посещаемость была наихудшей. Основными его задачами являются:

- составление расписания в журнале;
- ведение списков подгрупп;
- учёт посещаемости студентов подгрупп;
- составление статистики пропусков.

Также в ходе анализа был выявлен одна из основных базовых единиц, используемых, при работе старосты – занятие.

Занятие — это урок в учебном заведении по определённой дисциплине, определённой длительности и в определённое время. Каждое занятие предполагает посещение его студентами той группы, у которой проводится это занятие. Отсутствие на занятии может быть по причине (болезнь, командировка и т.п.) или без неё, при этом в любом случае может быть проставлена оценка за отработку этого занятия. Если же студент присутствовал он также может получить оценку, смысловая нагрузка которой определяется преподавателем.

Учёт посещаемости на занятии имеет очень большое значение для эффективной проверки качества обучения студентов в университете по отдельным предметам и в целом. Статистика даёт возможность оценивать загруженность преподавателей и добросовестность студентов. Она также может быть использована с целью принятия решения о дисциплинарных взысканиях студента.

1.2 Анализ отношений между объектами

Исходя из того, что пользователями приложения будут старосты должна быть предусмотрена авторизация и разграничение доступа к закрытой информации (персональная информация студентов). Определим следующие функциональные требования для приложения:

- авторизация;
- хранение актуальной информации о занятиях;
- возможность добавления, удаления и редактирования данных о подгруппах и занятиях;
- возможность изменять список групп исключая и добавляя в него студентов;
- возможность учёта посещаемости студентов;
- интуитивно понятный интерфейс;
- предоставление общей информации о группе и студентах.

Согласно вышеперечисленным функциональным требованиям, выделим следующие сущности, которые будут описывать нашу систему: университет, институт, кафедра, дисциплина, преподаватель, студент, занятие, посещение, пользователь и группа.

Каждой из перечисленной сущности соответствует свой набор атрибутов:

- университет: идентификатор, название университета, его адрес, сайт, контактные телефон и почта;
- институт: идентификатор, название института, идентификатор университета, сайт и контактные телефон и почта;
- кафедра: идентификатор, название кафедры, идентификатор института, сайт и контактные телефон и почта;
- дисциплина: идентификатор, название дисциплины;
- преподаватель: идентификатор, Ф.И.О., адрес проживания, контактные телефон и почта;

- студент: идентификатор, Ф.И.О., адрес проживания, контактные телефон и почта;
- занятие: идентификатор, идентификатор дисциплины, идентификатор преподавателя, идентификатор группы, кабинет, дата и время, длительность, тип.
- посещение: идентификатор, идентификатор занятия, идентификатор студента, оценка, присутствие, причина отсутствия.
- группа: идентификатор, название, курс, идентификатор кафедры, идентификатор старосты, идентификатор родительской группы;
- пользователь: идентификатор, имя, почта, пароль, идентификатор студента, ключ сессии.

2 РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ БАЗЫ ДАННЫХ

Для проектирования базы данных была выбрана реляционная модель. Такие модели позволяют достигнуть гораздо более высокого уровня абстракции, чем в сетевой или иерархической моделях. В основе реляционной модели лежит математическая теория отношений, имеющая строгие правила проектирования, включающие в себя необходимость проведения процедуры нормализации. Представление самих данных в модели не зависит от способа их физической реализации. Единственной используемой конструкцией является «таблица», что обеспечивает простоту и доступность для понимания пользователями различного уровня подготовки. Модели обеспечивают полную независимость данных, что подразумевает минимальные изменения в интерфейсе при внесении поправок в структуру модели. Стоит также отметить, что реляционные модели существенно сокращают избыточность и дублирование исходных данных, обеспечивают сохранение их целостности.

2.1 Построение диаграммы «сущность-связь» в нотации П.Чена

После того, как была проведен анализ предметной области, и выделены основные сущности, участвующие в работе отдела кадров университета, стало возможным построение диаграммы «сущность-связь» в нотации П.Чена, но для начала необходимо определить отношения между выделенными сущностями. К пользователю может быть привязан только один студент, так же как и к одному студенту может быть привязан только один пользователь – это отношение один к одному. У университета имеется множество институтов, у института имеется множество кафедр, у кафедры имеется множество групп, у преподавателя, по дисциплине и у группы имеется множество занятий, у занятий и студентов может быть множество посещений, это все отношения один ко многим. Так как студент может быть в нескольких подгруппах контрактов, а в одной подгруппе могут быть

несколько студентов, то мы имеем отношение многие ко многим. Также дисциплины могут преподаваться многими преподавателями и преподаватели могут вести несколько дисциплин, они же могут быть на нескольких кафедрах, в то же время на одной кафедре может быть больше одного преподавателя, потому что тоже отношение многие ко многим. Разработанная диаграмма представлена на рисунке А.1 приложения А.

2.2 Построение полной атрибутивной модели в нотации IDEF1X

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных. За основу возьмем выделенные ранее сущности с соответствующими им атрибутами.

Рассмотрим все выделенные нами сущности, и проверим их на атомарность. В каждой из сущности соблюдается атомарность, следовательно, делаем вывод о том, что наша база данных находится в первой нормальной форме.

Так как наша база данных, находится в первой нормальной форме, рассмотрим её на удовлетворение второй нормальной форме (Отношение R находится во второй нормальной форме в том и только в том случае, когда находится в 1НФ, и каждый не ключевой атрибут полностью зависит от первичного ключа). Каждая сущность содержит уникальный, искусственно введенный, первичный ключ, не обладающий смысловой нагрузкой, от которого полностью зависят все не ключевые атрибуты, это удовлетворяет условиям второй нормальной форме, следовательно, делаем вывод о том, что база данных находится во второй нормальной форме.

Так как наша база данных находится во второй нормальной форме, рассмотрим на её удовлетворение третьей нормальной форме (Отношение R находится в третьей нормальной форме в том и только в том случае, если находится в 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа). Во всех созданных сущностях транзитивные зависимости отсутствуют, что говорит о том, что наша база данных находится в третьей нормальной форме.

В ходе нормализации была построена полная атрибутивная модель в нотации IDEF1X, которая представлена на рисунке 2.2.

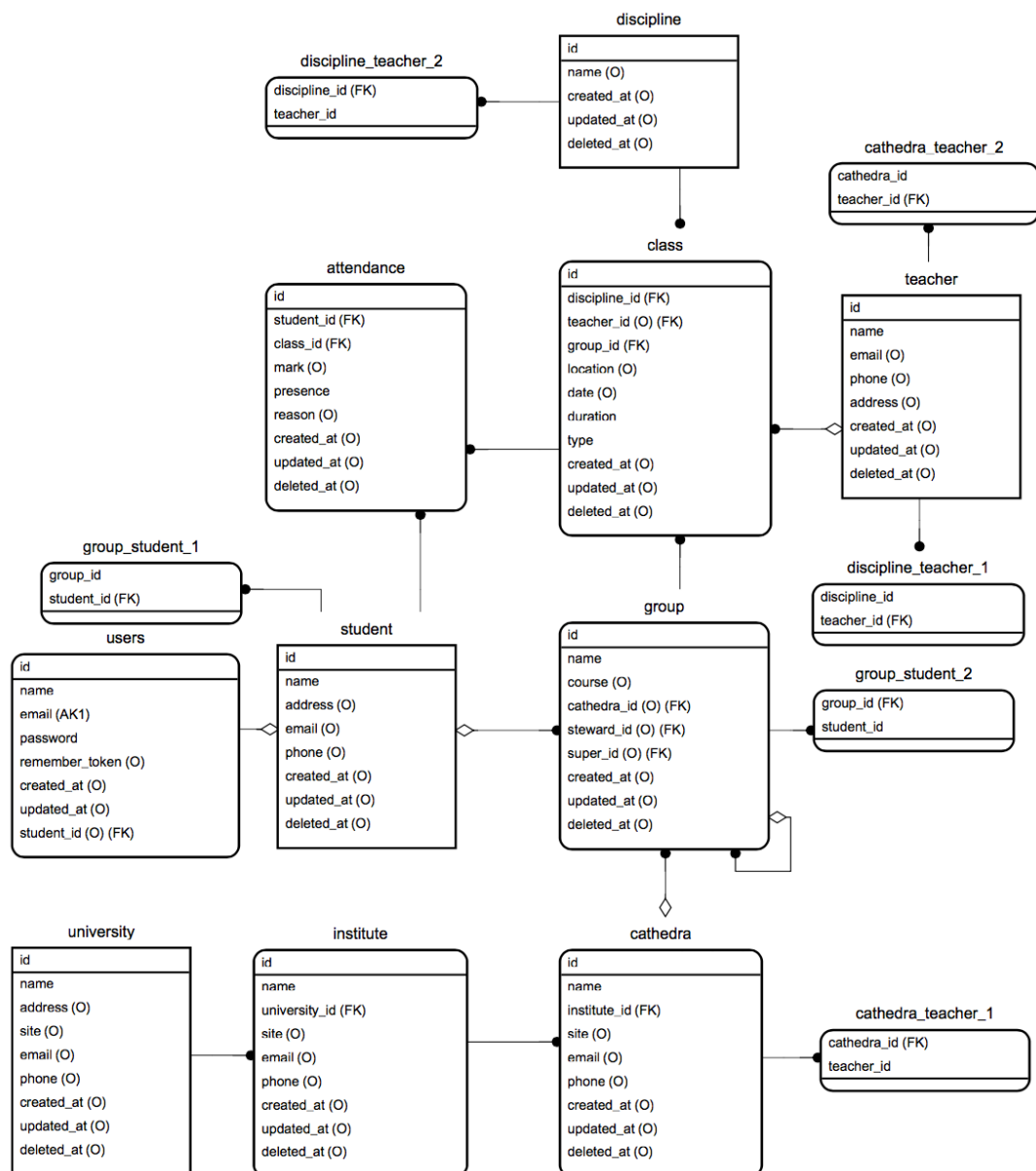


Рисунок 2.2 – Полная атрибутивная модель в нотации IDEF1X

3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ

3.1 Обоснование выбора СУБД

Выбор системы управления базами данных (СУБД) представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений баз данных. Выбранная СУБД должна удовлетворять как текущим, так и будущим потребностям системы. В настоящее время существует достаточно много различных СУБД, но в данной курсовой работе, была использована *MySQL*. Так как при разработке используется язык программирования *PHP* и фреймворк *Laravel 5*, содержащий свой модуль для работы с базой данных *Eloquent*, который поддерживает различные СУБД выбор был сделан на основе личного опыта использования СУБД *MySQL*, а также простоты её применения. Эксплуатация системы будет производиться с любой операционной системы в любом современном браузере.

3.2 Разработка физической модели базы данных

Так как у нас уже имеется логическая модель базы данных в третьей нормальной форме, а также выбрана СУБД, которая её будет реализовать, преобразуем логическую модель в физическую слиянием, заранее разделённых, таблиц развязки отношений многие ко многим. На рисунке 3.1 представлена физическая модель нашей базы данных.

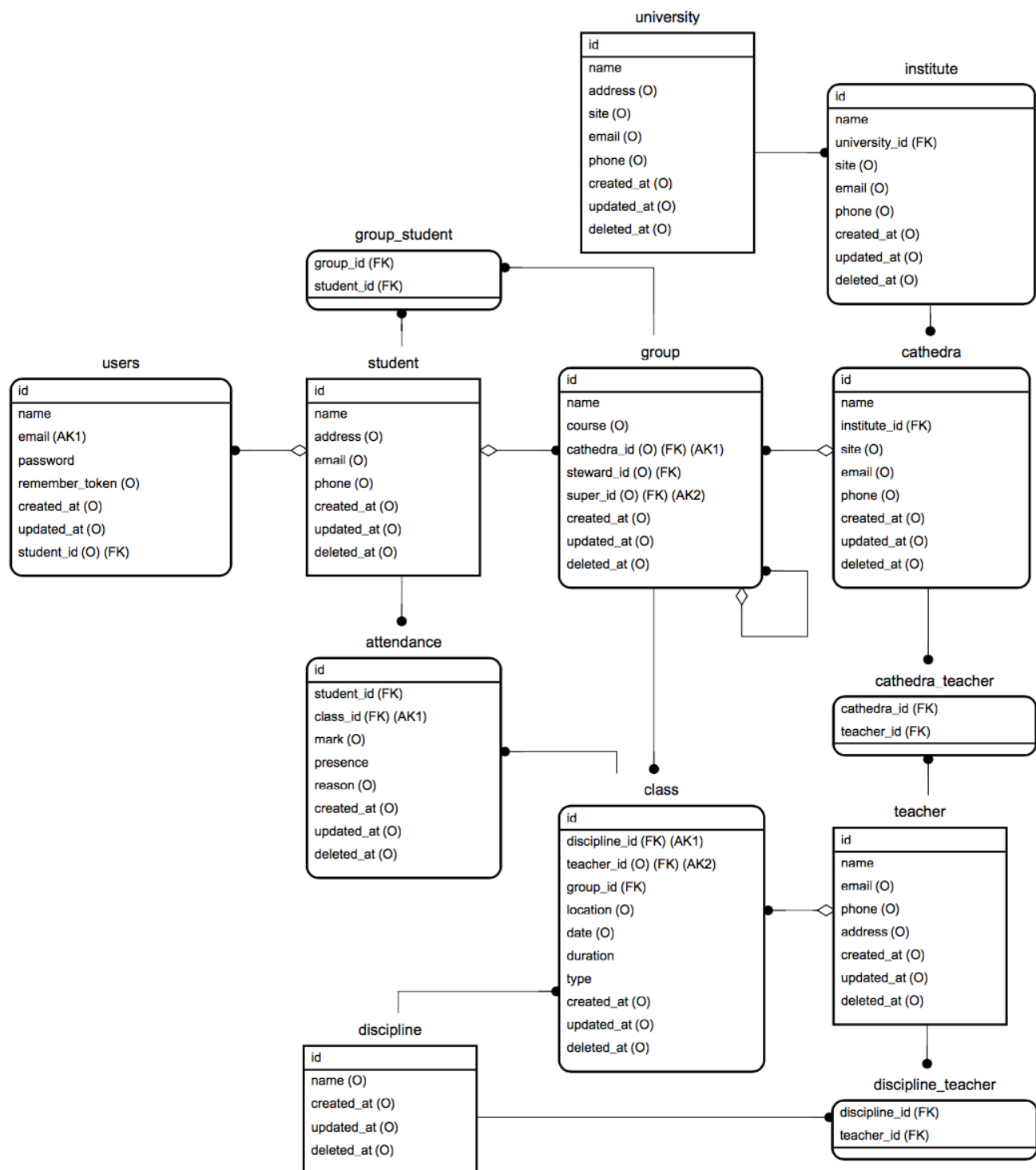


Рисунок 3.1 – Физическая модель базы данных

3.3 Разработка запросов к базе данных

При анализе предметной области, были разработаны функциональные требования к базе данных, на основании которых, была разработана логическая и физическая модель базы данных.

В рамках данной курсового проекта работа с базой данных осуществляется с использованием вышеупомянутого фреймворка (*Laravel Eloquent*), в котором был использован подход миграций, который позволяет определять специальные методы, создающие и изменяющие таблицы в базе данных. Цепочка таких миграций позволит на любом конечном сервере установить систему за пару минут. Так же в каждую таблицу (кроме развязочных) добавлены служебные временные метки («*created_at*», «*updated_at*» и в некоторых случаях «*deleted_at*»), являющиеся полями типа *TIMESTAMP* и хранящие соответственно время добавления записи, её последнего редактирования и удаления. Последняя метка используется для реализации подхода *SoftDelete*, который позволяет реализовать «псевдо»-удаления путём выставления текущей временной метки в поле. Такой подход позволяет решить проблему случайного удаления, позволяя администратору базы данных восстановить ошибочно удалённые данные, не обращаясь в резервным копиям.

Исходя из функциональных требований, запросы к базе данных, должны реализовываться:

- студентов в конкретной группе или подгруппе;
- выборку университетов, институтов, кафедр, дисциплин и преподавателей с определенными условиями сужения выборки;
- подсчет суммарного количества пропусков студентов.

Данные запросы реализуется с использованием обращений к методам моделей *Eloquent ORM*.

3.4 Разграничение прав доступа

В базе данных подобного типа существует необходимость многокритериального ограничения доступа на уровне строк, но на уровне СУБД *MySQL* единственным способом создания ограничения на уровне строк является создание представлений таблиц со специально сформированными условиями и триггерами. Из-за сложности и в некоторых деталях невозможности такого подхода разграничение прав доступа реализовано на прикладном уровне. Так как СУБД не доступна из внешней сети и может использоваться только системой, установленной непосредственно на том же сервере, который считается доверенным, имеется возможность предоставить системе полные права доступа к созданной базе данных.

3.5 Исследование информационных параметров базы данных

Длина логической записи j -ого файла определяется как сумма длин полей, её составляющих и указателей, если они организуются разработчиком

$$L_j = \sum_{i=1}^{M_j} l_{ij} \quad [\text{байт}], \quad (3.1)$$

где M_j – число групп полей в записях;

l_{ij} длина группы [байт].

Вычислим:

$$L_{\text{university}} = 4 + 5 * (255 + 1) + 3 * 4 = 1\,296 \text{ байт.}$$

$$L_{\text{institute}} = 4 + 4 * (255 + 1) + 4 + 3 * 4 = 1\,044 \text{ байта.}$$

$$L_{\text{cathedra}} = 4 + 4 * (255 + 1) + 4 + 3 * 4 = 1\,044 \text{ байта.}$$

$$L_{\text{discipline}} = 4 + 1 * (64 + 1) + 3 * 4 = 81 \text{ байт.}$$

$$L_{\text{teacher}} = 4 + 4 * (255 + 1) + 3 * 4 = 1\,040 \text{ байт.}$$

$$L_{\text{group}} = 4 + 1 * (64 + 1) + 4 * 4 + 3 * 4 = 97 \text{ байт.}$$

$$L_{\text{student}} = 4 + 4 * (255 + 1) + 3 * 4 = 1\,040 \text{ байт.}$$

$$L_{\text{class}} = 4 + 1 * (255 + 1) + 3 * 4 + 5 + 1 + 3 + 3 * 4 = 293 \text{ байта.}$$

$$L_{\text{attendance}} = 4 + 3 * 4 + 1 + 1 + 3 * 4 = 30 \text{ байт.}$$

$$L_{\text{group_student}} = 4 + 4 = 8 \text{ байт.}$$

$$L_{\text{discipline_teacher}} = 4 + 4 = 8 \text{ байт.}$$

$$L_{\text{cathedral_teacher}} = 4 + 4 = 8 \text{ байт.}$$

Объем памяти, необходимый для размещения информационного фонда без учёта системных данных и указателей составит

$$I = \sum_{j=1}^N m\{L_j\}K_j \quad [\text{байт}], \quad (3.2)$$

где N – число типов записей в информационном фонде;

K_j – количество записей j -го файла.

Предположим, что база содержит:

1 университет, 1 институт, 1 кафедру, 11 дисциплин, 14 преподавателей, 6 групп, 24 студента, 265 занятий, 1 125 посещений, 95 связей группа-студент, 19 связей дисциплина-преподаватель и 14 связей кафедра-преподаватель.

$$I = 1\,296 * 1 + 1\,044 * 1 + 1\,044 * 1 + 81 * 11 + 1\,040 * 14 + 97 * 6 + 1\,040 * 24 + 293 * 265 + 30 * 1\,125 + 8 * 95 + 8 * 19 + 8 * 14 = 278\,818 \text{ (байт).}$$

Приращение информационного фонда

$$\Delta I = \sum_{j=1}^N m\{L_j\}\Delta\lambda_j \quad [\text{байт}^{-1}], \quad (3.3)$$

где N – число добавленных типов записей

λ_j – интенсивность добавления записей в файл j -го типа.

$$\begin{aligned} \Delta I = & 1296 * \frac{1}{1125} + 1044 * \frac{1}{1125} + 1044 * \frac{1}{1125} + 81 * \frac{11}{1125} + 1040 * \frac{14}{1125} \\ & + 97 * \frac{6}{1125} + 1040 * \frac{24}{1125} + 293 * \frac{265}{1125} + 1296 * \frac{1}{1125} + 1044 \\ & * \frac{1}{1125} + 1044 * \frac{1}{1125} + 81 * \frac{11}{1125} + 1040 * \frac{14}{1125} + 97 * \frac{6}{1125} \\ & + 1040 * \frac{24}{1125} + 293 * \frac{265}{1125} + 30 * \frac{1125}{1125} + 8 * \frac{95}{1125} + 8 * \frac{19}{1125} \\ & + 8 * \frac{14}{1125} = 247,83 \text{ (байт}^{-1}\text{)}. \end{aligned}$$

Время заполнения информационного фонда

$$I_{\text{зАП}} = \frac{(V_{\text{общ}} - V_{\text{по}} - I)}{\Delta I} \quad [\text{время}] \quad (3.4)$$

$$I_{\text{зАП}} = \frac{5\,368\,709\,120 - 356\,410\,982,4 - 278\,818}{247,83} = 20\,223\,618 \text{ (дней)}$$

Время резервного копирования определяется интенсивностью отказов, сопровождающихся потерей данных

$$T_{\text{рк}} = \frac{0,2 * I}{\Delta I} = \frac{0,2 * 278\,818}{247,83} = 225,007464794$$

4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

4.1 Обоснование выбора языка программирования

При написания приложения для данного курсового проекта был выбран язык программирования *PHP*. Этот выбор обусловлен тем, что данный язык позволяет применять объектно-ориентированный подход, кроссплатформенен и имеет развитое сообщество, также что не мало важно – сам интерпретатор *PHP* и множество *HTTP*-серверов под него являются бесплатными.

Приложение написано с помощью среды разработки *JetBrains PHPStorm* 2016.3 с использованием фреймворка *Laravel 5*. Этот фреймворк выбран за высокий уровень безопасности и удобства работы с ним, так как имеет множество реализованных функций, позволяющих упростить создание приложений. Также в ходе работы была использована распределённая система управления версиями – *GitHub*. Для подключения и реализации работы с базой данных использовался *Eloquent ORM*, входящий в состав фреймворка *Laravel 5*.

4.2 Разработка интерфейса пользователя

Как упоминалось выше, для создания приложения использовался фреймворк *Laravel 5*, для визуального отображения интерфейса был выбран также фреймворк *Bootstrap 3*, позволяющий быстро создать удобный интерфейс пользователя. Были спроектированные следующие страницы: приветствие, расписание на сегодня, просмотр, добавление, редактирование и удаление групп, изменение списка студентов и статистика пропусков группы, просмотр, добавление, редактирование, удаление занятий, а также служебные страницы регистрации, авторизации и восстановления пароля. Также во всех списках сущностей, доступ к редактированию которых имеется у пользователя, появляется

соответствующие кнопки редактирования и удаления, а в случае со списками студентов поля появляются сразу рядом со списком.

Для каждого поля ввода данных была добавлена валидация, при удалении записи происходит переход на страницу, предупреждающую об удалении. Для каждой сущности реализованы страницы для редактирования и просмотра общей информации. Далее будут рассмотрены все перечисленные страницы на рисунках.

На рисунке 4.1 представлен скриншот приложения демонстрирующий страницу приветствия.

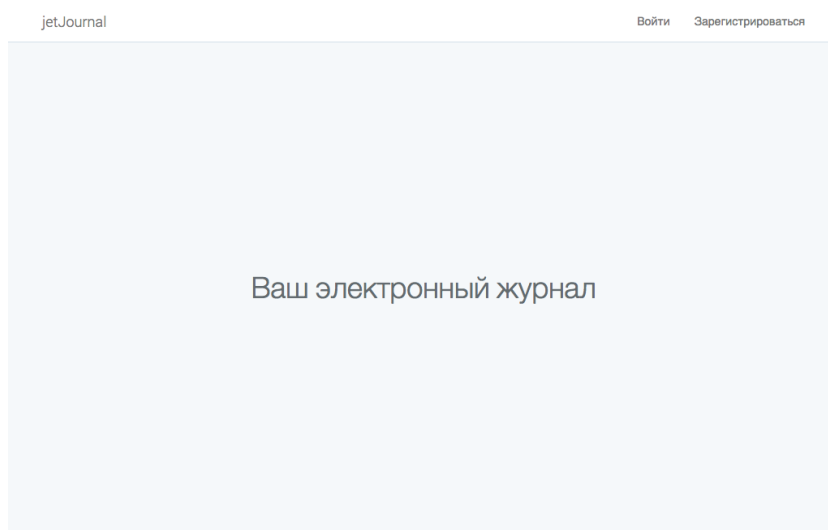


Рисунок 4.1 – Окно авторизации

На рисунке 4.2 представлен скриншот страницы авторизации.

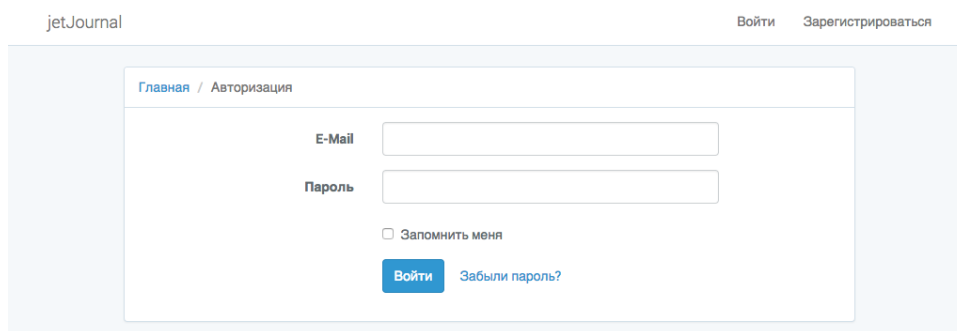


Рисунок 4.2 – Страница авторизации

На рисунке 4.3 представлен скриншот страницы занятий на сегодня.

jetJournal Группы Занятия Курчки Ариф ▾

Главная		
Занятия на сегодня 09.12.2016 (Пт)		
11 ⁵⁰ – 13 ²⁰ 📍 Б-501	Лабораторная работа Инструментальные средства информационных систем	Карлусов В. Ю.
11 ⁵⁰ – 13 ²⁰ 📍 Б-502	Лабораторная работа Методы и средства хранения информации	Балясный Н. В.
14 ⁰⁰ – 15 ³⁰ 📍 Б-507	Лабораторная работа Тестирование ПО	Строганов В. А.
15 ⁴⁰ – 17 ¹⁰ 📍 Г-511	Семинар Стандартизация, сертификация и унификация ИТ	Кузнецов С. А.

Рисунок 4.3 – Страница занятий на сегодня

На рисунке 4.3 представлены все занятия текущего пользователя на сегодня, зелёным отмечаются прошедшие занятия, красным идущие, а жёлтым предстоящие.

На рисунке 4.4 представлен скриншот страницы списка групп.

jetJournal Группы Занятия Курчки Ариф ▾

Главная / Группы		+			
ID	Название				
1	ИС/Б-3-о				
2	ИС/Б-31-о	📊	📋	✎	🗑
3	ИС/Б-31-о ИСИС-1	📊	📋	✎	🗑
4	ИС/Б-31-о ИСИС-2	📊	📋	✎	🗑
5	ИС/Б-31-о МисХИ	📊	📋	✎	🗑
6	ИС/Б-31-о ОРИП	📊	📋	✎	🗑

Рисунок 4.4 – Страница списка групп

На представленном рисунке (4.4) группы, старостой которых является текущий пользователь имеют соответствующие управляющие кнопки (статистика, редактирование списка, редактирование группы и её удаление), а верхняя кнопка в заголовке панели позволяет создать новую группу.

На рисунке 4.5 представлен скриншот страницы информации о группе.

























Главная / Группы / ИС/6-31-о			
Севастопольский Государственный Университет			
Информационных технологий и управления в технических системах			
Информационных систем			
ID	ФИО	E-mail	Телефон
1	 Акименко Владислав Андреевич		+7 (978) 800-73-48
2	 Астахов Артем Сергеевич		+7 (978) 850-19-47
3	 Баранов Артем Юрьевич		+7 (978) 039-37-30
4	 Бордиян Ирина Игоревна		+7 (978) 705-92-58
5	 Бородаченко Денис Викторович		+7 (978) 848-62-95
6	 Волков Андрей Алексеевич	wolfain@mail.ru	+7 (978) 809-48-02
7	 Демидова Надежда Андреевна	naday.demidowa@gmail.com	+7 (978) 785-19-92
8	 Ковганов Михаил Олегович		+7 (978) 710-31-52
9	 Корж Алексей Игоревич		+7 (978) 027-21-17
10	 Коротков Иван Викторович		+7 (978) 074-85-28
11	 Курчки Ариф Эрнестович	justnero.ru@yandex.ru	+7 (978) 808-44-64
12	 Мазур Татьяна Андреевна	tanyamazur95@gmail.com	+7 (978) 779-32-71
13	 Мжачев Илья Александрович	mzhaches@mail.ru	+7 (989) 082-72-99
14	 Ордин Роман Владимирович		+7 (978) 772-69-20
15	 Оrobeц Андрей Александрович		+7 (918) 653-41-05
16	 Повх Андрей Анатольевич		+7 (978) 888-05-73
17	 Салюк Игорь Юрьевич		+7 (978) 745-89-43
18	 Степанова Елизавета Владимировна		+7 (978) 048-79-29
19	 Степанян Роберт Леонович		+7 (978) 754-73-93
20	 Струшкевич Данил Александрович		+7 (989) 232-27-55
21	 Сухой Дмитрий		
22	 Таушканов Герман Владимирович		+7 (978) 032-68-90
23	 Хицун Светлана Игоревна		+7 (978) 061-86-54
24	 Юрчик Александр Сергеевич		+7 (978) 852-09-53

Рисунок 4.5 – Информация о группе

На представленном рисунке (4.5) присутствует текущий список группы, в случае если текущий пользователь является старостой этой группы он подсвечивается зелёным, выводится дополнительная информация о студентах, а также управляющие кнопки в заголовке панели. Иконка карточки слева от имени каждого студента ведёт на его персональную страницу, доступную только старосте и самому студенту.

На рисунке 4.6 представлен скриншот страницы информации о пользователе.

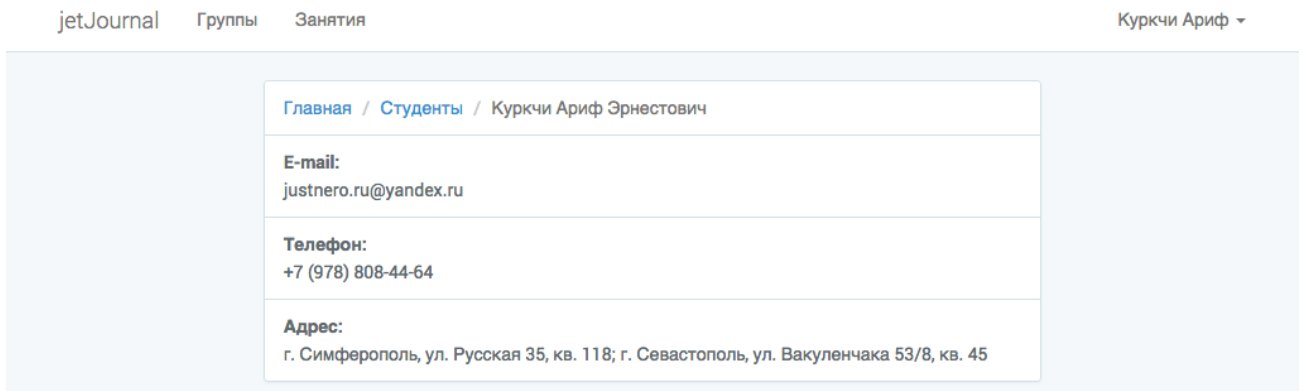


Рисунок 4.6 – Страница информации о пользователе

На рисунке 4.7 представлен скриншот страницы списка занятий.

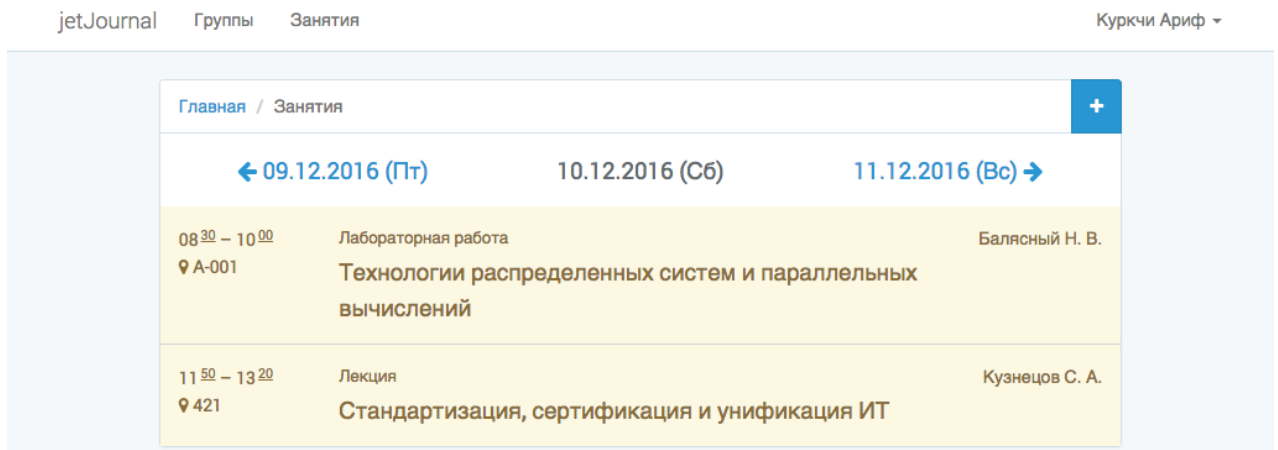


Рисунок 4.7 – Страница списка занятий

На представленном рисунке (4.7) для перехода между датами служат соответствующие ссылки слева и справа от текущей выбранной даты. Для старост отображается дополнительная кнопка создания занятий в заголовке панели.

На рисунке 4.8 представлен скриншот страница занятия с редактированием списка присутствующих.

Главная / Занятия

09.12.2016 (Пт)

15:40 – 17:10

Семинар

Кузнецов С. А.

Г-511

Стандартизация, сертификация и унификация ИТ

01 . Акименко Владислав Андреевич	<input checked="" type="checkbox"/>	Оценка
02 . Астахов Артем Сергеевич	<input type="checkbox"/>	Причина
03 . Баранов Артем Юрьевич	<input checked="" type="checkbox"/>	Оценка
04 . Бордиян Ирина Игоревна	<input checked="" type="checkbox"/>	Оценка
05 . Бородаченко Денис Викторович	<input type="checkbox"/>	Причина
06 . Волков Андрей Алексеевич	<input checked="" type="checkbox"/>	Оценка
07 . Демидова Надежда Андреевна	<input checked="" type="checkbox"/>	Оценка
08 . Ковганов Михаил Олегович	<input type="checkbox"/>	Причина
09 . Корж Алексей Игоревич	<input type="checkbox"/>	Причина
10 . Коротков Иван Викторович	<input type="checkbox"/>	Причина
11 . Куркчи Ариф Эрнестович	<input checked="" type="checkbox"/>	Оценка
12 . Мазур Татьяна Андреевна	<input checked="" type="checkbox"/>	Оценка
13 . Мжачев Илья Александрович	<input checked="" type="checkbox"/>	Оценка
14 . Ордин Роман Владимирович	<input type="checkbox"/>	Причина
15 . Оrobeц Андрей Александрович	<input type="checkbox"/>	Причина
16 . Повх Андрей Анатольевич	<input type="checkbox"/>	Причина
17 . Салюк Игорь Юрьевич	<input type="checkbox"/>	Причина
18 . Степанова Елизавета Владимировна	<input type="checkbox"/>	Причина
19 . Степанян Роберт Левонович	<input type="checkbox"/>	Причина
20 . Струшкевич Данил Александрович	<input type="checkbox"/>	Причина
21 . Сухой Дмитрий	<input type="checkbox"/>	Причина
22 . Таушканов Герман Владимирович	<input type="checkbox"/>	Причина
23 . Хицун Светлана Игоревна	<input checked="" type="checkbox"/>	Оценка
24 . Юрчик Александр Сергеевич	<input type="checkbox"/>	Причина

Сохранить

Рисунок 4.8 – Страница занятия

На представленном рисунке (4.8) выведена информация о занятии, а так же продублирован список группы, для которой это занятие проводится. Поля редактирования и соответственно кнопка сохранения доступна только для

старосты группы. Галочка обозначает присутствие студента на занятии, в случае её отсутствия отображается выпадающий список причины отсутствия. Поле оценки отображается в любом случае.

На рисунке 4.9 представлен скриншот страницы добавления группы.

jetJournal Группы Занятия Курчки Ариф ▾

Главная / Группы / Добавление

Название группы

Номер курса

Родительская группа

ИС/6-3-о ▾

Университет

Севастопольский Государственный Университет ▾

Институт

Информационных технологий и управления в технических системах ▾

Кафедра

Информационных систем ▾

Сохранить

Рисунок 4.9 – Страница добавления группы

На представленном рисунке (4.9) присутствуют 4 выпадающих списка. В первом необходимо выбрать родительскую группу, после этого появляются остальные. Если выбрана конкретная группа университет, институт и кафедра берутся из неё, иначе поля появляются по очереди, позволяя выбрать из доступных в базе значений. Страница редактирования группы аналогична.

На рисунке 4.10 представлен скриншот страницы добавления занятия.

jetJournal Группы Занятия Куркчи Ариф ▾

Главная / Занятия / Добавление

Группа
ИС/6-31-о ▾

Дисциплина
Инструментальные средства информационных систем ▾

Преподаватель
Балясный Н. В. ▾

Дата
09.12.2016, 22:49

Длительность
01:30

Кабинет

Тип занятия
Лекция ▾

Повторять
Нет ▾

Сохранить

Рисунок 4.10 – Страница добавления занятия

На представленном рисунке (4.10) присутствуют выпадающие списки групп (только тех, для которых текущий пользователь является старостой), дисциплин, преподавателей и типа занятия. Поле повтора позволяет продублировать занятие раз в неделю или в две до определённой даты.

На рисунке 4.11 представлен скриншот страницы статистики по пропускам.

Главная / Группы / ИС/6-31-о / Статистика пропусков																			
Топ	Студент		Недели															Всего	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
17	Акименко Владислав Андреевич	Без ув.	4	4	10	2	0	0	0	0	0	0	0	0	0	2	0	22	
		Всего	4	4	10	2	0	0	0	0	0	0	0	0	0	2	0	22	
5	Астахов Артем Сергеевич	Без ув.	12	20	24	4	0	0	0	0	2	0	0	0	0	2	2	66	
		Всего	12	20	24	4	0	0	0	0	2	0	0	0	0	2	2	66	
4	Баранов Артем Юрьевич	Без ув.	4	24	26	6	0	2	2	2	2	2	2	0	0	0	0	72	
		Всего	4	24	26	6	0	2	2	2	2	2	2	0	0	0	0	72	
15	Бордиян Ирина Игоревна	Без ув.	2	2	16	6	2	0	0	0	0	0	0	0	0	0	0	28	
		Всего	2	2	16	6	2	0	0	0	0	0	0	0	0	0	0	28	
13	Бородаченко Денис Викторович	Без ув.	2	6	18	2	0	0	0	0	0	0	0	2	0	0	2	32	
		Всего	2	6	18	2	0	0	0	0	0	0	0	2	0	0	2	32	
7	Волков Андрей Алексеевич	Без ув.	0	0	10	2	0	0	0	0	2	0	0	0	0	2	0	16	
		Всего	10	20	10	2	0	0	0	0	2	0	0	0	0	2	0	46	
24	Демидова Надежда Андреевна	Без ув.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		Всего	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	Ковганов Михаил Олегович	Без ув.	14	22	26	8	2	2	2	2	2	2	2	2	2	2	4	94	
		Всего	14	22	26	8	2	2	2	2	2	2	2	2	2	2	4	94	
20	Корж Алексей Игоревич	Без ув.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	
		Всего	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	
8	Коротков Иван Викторович	Без ув.	6	6	14	6	2	0	0	0	2	2	2	0	2	2	2	46	
		Всего	6	6	14	6	2	0	0	0	2	2	2	0	2	2	2	46	
22	Курчки Ариф Эрнестович	Без ув.	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	
		Всего	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	
19	Мазур Татьяна Андреевна	Без ув.	0	2	6	2	0	0	0	0	0	0	0	0	0	0	0	10	
		Всего	0	2	6	2	0	0	0	0	0	0	0	0	0	0	0	10	
23	Мжачев Илья Александрович	Без ув.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		Всего	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	Ордин Роман Владимирович	Без ув.	14	22	26	8	2	2	2	2	2	2	2	2	2	2	4	94	
		Всего	14	22	26	8	2	2	2	2	2	2	2	2	2	2	4	94	
11	Оробец Андрей Александрович	Без ув.	4	14	16	4	0	0	0	0	0	0	0	0	0	0	2	40	
		Всего	4	14	16	4	0	0	0	0	0	0	0	0	0	0	2	40	
21	Повх Андрей Анатольевич	Без ув.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	
		Всего	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	
18	Салюк Игорь Юрьевич	Без ув.	4	6	8	0	0	0	0	0	0	0	0	0	0	0	2	20	
		Всего	4	6	8	0	0	0	0	0	0	0	0	0	0	0	2	20	
10	Степанова Елизавета Владимировна	Без ув.	2	14	12	2	0	0	0	2	0	2	0	0	2	2	4	42	
		Всего	2	14	12	2	0	0	0	2	0	2	0	0	2	2	4	42	
9	Степанян Роберт Леонович	Без ув.	12	10	10	4	0	0	0	0	2	0	0	0	2	2	2	44	
		Всего	12	10	10	4	0	0	0	0	2	0	0	0	2	2	2	44	
14	Струшкевич Данил Александрович	Без ув.	4	2	10	4	2	0	0	0	0	0	0	2	0	0	2	26	
		Всего	4	2	10	4	2	0	0	0	0	2	0	2	0	0	2	28	
3	Сухой Дмитрий	Без ув.	14	22	26	8	2	2	2	2	2	2	2	2	2	2	4	94	
		Всего	14	22	26	8	2	2	2	2	2	2	2	2	2	2	4	94	
16	Таушканов Герман Владимирович	Без ув.	4	2	10	4	0	0	0	0	0	0	0	0	2	2	2	24	
		Всего	4	2	10	4	0	0	0	0	0	2	0	0	0	2	2	26	
6	Хицун Светлана Игоревна	Без ув.	2	6	24	8	2	0	2	0	0	0	0	2	2	2	2	52	
		Всего	2	6	24	8	2	0	2	0	0	0	0	2	2	2	2	52	
12	Юрчик Александр Сергеевич	Без ув.	4	10	16	2	0	0	0	0	0	0	0	0	0	0	2	34	
		Всего	4	10	16	2	0	0	0	0	0	0	0	0	0	0	2	34	

Рисунок 4.11 – Страница статистики по пропускам

На представленном рисунке (4.11) выведена таблица всех студентов группы, красным подсвечены 5 лидеров по количеству пропусков. Для каждой недели выводится по 2 значения количество пропусков без уважительной причины и общее количество пропусков в академических часах. При наведении на номер недели высвечиваются даты этой недели.

На рисунках продемонстрированы основные страницы приложения. Страницы удаления являются элементарными и содержат дубликат информации о записи и кнопку подтверждения удаления. Страницы редактирования аналогичны страницам создания, но с некоторыми заблокированными полями.

4.3 Тестирование работы приложения

Произведем вывод статистики по пропускам студентов группы ИС/б-31-о. Результаты представлены на рисунке 4.11 выше (выборка приложения) и рисунке 4.12 (выборка написанного вручную запроса в СУБД).

```

SELECT student.name as "Студент", WEEK(class.date, 1) - 34 as "Неделя",
((COUNT(attendance.presence)-SUM(attendance.presence)) as 'Пар', ((COUNT(attendance.presence)-SUM(attendance.presence))*2) as "Часов"
FROM attendance
JOIN student
ON attendance.student_id = student.id
JOIN class
ON attendance.class_id = class.id
WHERE attendance.student_id IN (SELECT group_student.student_id FROM group_student WHERE group_student.group_id = 2) AND
WEEK(class.date, 1) = 35
GROUP BY attendance.student_id, WEEK(class.date, 1);

```

Студент	Неделя	Пар	Часов
1. Акименко Владислав Андреевич	1	2	4
2. Астахов Артем Сергеевич	1	6	12
3. Баранов Артем Врьевич	1	2	4
4. Бордюин Ирина Игоревна	1	1	2
5. Бородаченко Денис Викторович	1	1	2
6. Волков Андрей Алексеевич	1	5	10
7. Демидова Надежда Андреевна	1	0	0
8. Ковганов Михаил Олегович	1	7	14
9. Корж Алексей Игоревич	1	0	0
10. Коротков Иван Викторович	1	3	6
11. Курочки Ариф Эрнестович	1	0	0
12. Мазур Татьяна Андреевна	1	0	0
13. Мичаев Илья Александрович	1	0	0
14. Ордин Роман Владимирович	1	7	14
15. Оробец Андрей Александрович	1	2	4
16. Павлов Андрей Анатольевич	1	0	0
17. Салок Игорь Врьевич	1	2	4
18. Степанова Елизавета Владимировна	1	1	2
19. Степанян Роберт Леонович	1	6	12
20. Стружневич Данил Александрович	1	2	4
21. Сухой Дмитрий	1	7	14
22. Таууканов Герман Владимирович	1	2	4
23. Хищин Светлана Игоревна	1	1	2
24. Врич Александр Сергеевич	1	2	4

Рисунок 4.12 – Выборка запросом в СУБД

Исходя из рисунков 4.11 и 4.12 делаем вывод, о том, что приложение работает корректно.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была спроектирована реляционная база данных на тему «База данных электронного журнала старосты». Написано приложение на языке *PHP*, реализующее работу с базой данных. Также был разработан простой и изящный интерфейс для работы с базой данных. Кроме того, была обеспечена защита базы данных, путём разграничение прав доступа по строкам на прикладном уровне.

Для спроектированной логической модели базы данных была проведена нормализация до третьей нормальной формы, которая позволила уменьшить избыточность данных, а также сохранить целостность данных. Реализована физическая модель базы данных.

Были закреплены навыки проектирования логической модели базы данных, проведения их нормализации, построения физической модели, разработки программного обеспечения, использующие базы данных как способ хранения данных и был изучен фреймворк *Laravel 5* и *Eloquent ORM* для работы с базами данных и создания удобных веб-приложений.

В результате выполнения курсового проекта был получен программный продукт, реализующий электронный журнал старосты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гарсиа-Молина Г. Системы баз данных. Полный курс: Пер. с англ./Г.Гарсиа-Молина, Д.Д.Ульман, Д.Уидом.— М.: Издательский дом «Вильямс», 2004.—1088с.
2. Дейт К.Дж. Введение в системы баз данных, 7-е издание.:Пер. с англ./К.Дж.Дейт.— М.: Издательский дом «Вильямс», 2002.—1072с.
3. Ульман Д. Основы систем баз данных/Д.Ульман.— М.: Финансы и статистика, 1983. — 334 с.

ПРИЛОЖЕНИЕ А



Рисунок А.1 – Диаграмма в нотации П. Чена

ПРИЛОЖЕНИЕ Б

Текст программы

Текст контроллера главной страницы:

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Clazz;
6  use App\Student;
7  use Carbon\Carbon;
8
9  class HomeController extends Controller
10 {
11     /**
12      * Create a new controller instance.
13      */
14     public function __construct()
15     {
16         $this->middleware('auth', ['except' => ['index']]);
17     }
18
19     /**
20      * Show the application dashboard.
21      *
22      * @return \Illuminate\Http\Response
23      */
24     public function index()
25     {
26         if(auth()->guest()) {
27             return view('home.welcome');
28         }
29
30         /* @var Student $student */
31         $student = auth()->user()->student;
32
33         $today = Carbon::now()->startOfDay();
34         $groups = array_map(function ($el) {
35             return $el->id;
36         }, $student->groups()->get(['id']->all());
37         $classes = Clazz::whereIn('group_id', $groups)
38             ->whereBetween('date', [$today, $today->copy()->endOfDay()])
39             ->with('discipline', 'teacher')
40             ->ordered()
41             ->get()
42             ->all();
43
44         $classStatus = function (Clazz $class) {
45             $now = Carbon::now();
46             if($now->lt($class->date)) {
47                 return 'future';
48             }
49             if($now->lte($class->duration)) {
50                 return 'present';
51             }
52             return 'past';
53         };
54
55         return view('home.dashboard', compact('classes', 'today', 'classStatus'));
56     }
57 }
```

Текст контроллера студентов:

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Student;
6  use Illuminate\Http\Request;
7  use Symfony\Component\Finder\Exception\AccessDeniedException;
```

```

8
9 class StudentController extends Controller
10 {
11
12     public function show(Student $student) {
13         /* @var 'Student $student */
14         $me = auth()->user()->student;
15         $allowed = true;
16         if($student->id != $me->id) {
17             $allowed = false;
18             foreach ($student->groups as $group) {
19                 if($group->steward_id == $me->id) {
20                     $allowed = true;
21                     break;
22                 }
23             }
24         }
25         if(!$allowed) {
26             abort(403, 'Для того, что бы зайти на эту страницу необходимо быть старостой');
27         }
28
29         return view('student.show', compact('student'));
30     }
31
32     public function link() {
33         $me = auth()->user();
34         if(!$me->student_id) {
35             return back();
36         }
37     }
38 }
39
40 }

```

Текст контроллера групп:

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Attendance;
6 use App\Clazz;
7 use App\Group;
8 use App\Institute;
9 use App\Student;
10 use App\University;
11 use Illuminate\Http\Request;
12 use Illuminate\Validation\Rule;
13
14 class GroupController extends Controller
15 {
16     /**
17      * Create a new controller instance.
18      */
19     public function __construct()
20     {
21         $this->middleware(['auth', 'student']);
22         $this->middleware(['steward', ['only' => ['edit', 'delete', 'update', 'destroy']]);
23     }
24
25     public function index()
26     {
27         /* @var Student $student */
28         $student = auth()->user()->student;
29         $groups = [];
30         foreach ($student->groups()->select(['id', 'name', 'steward_id'])->get()->all() as $group) {
31             $groups[$group->id] = $group;
32         }
33         foreach ($student->steward()->select(['id', 'name', 'steward_id'])->get()->all() as $group) {
34             $groups[$group->id] = $group;
35         }
36         ksort($groups);
37         return view('group.index', compact('groups', 'student'));
38     }
39
40     public function create(Request $request)
41     {
42         if ($request->ajax()) {
43             return $this->create_ajax($request);
44         }
45     }
46 }

```

```

45
46     $group = new Group();
47     $supers = [];
48     $student = auth()->user()->student;
49     $supers[null] = 'HET';
50     foreach (Group::roots()->all() as $el) {
51         $supers[$el->id] = $el->name;
52     }
53     foreach ($student->stewarded->all() as $el) {
54         $supers[$el->id] = $el->name;
55     }
56     $universities = [
57         null => 'Выберите университет'
58     ];
59     foreach (University::all() as $university) {
60         $universities[$university->id] = $university->name;
61     }
62
63     return view('group.create', compact('group', 'supers', 'universities'));
64 }
65
66 private function create_ajax(Request $request)
67 {
68     if ($request->has('super_id')) {
69         $super = Group::find($request->get('super_id'));
70         if ($super) {
71             return [
72                 'university_id' => $super->cathedra->institute->university_id,
73                 'institute' => $super->cathedra->institute->name,
74                 'cathedra' => $super->cathedra->name,
75             ];
76         }
77         abort(404);
78     } else if ($request->has('university_id')) {
79         $university = University::find($request->get('university_id'));
80         if ($university) {
81             $list = [];
82             foreach ($university->institutes as $el) {
83                 $list[$el->id] = $el->name;
84             }
85             return $list;
86         }
87         abort(404);
88     } else if ($request->has('institute_id')) {
89         $institute = Institute::find($request->get('institute_id'));
90         if ($institute) {
91             $list = [];
92             foreach ($institute->cathedras as $el) {
93                 $list[$el->id] = $el->name;
94             }
95             return $list;
96         }
97         abort(404);
98     }
99 }
100
101 public function show(Group $group)
102 {
103     $student_id = auth()->user()->student->id;
104     $steward_id = $group->steward_id;
105     $canEdit = $steward_id == $student_id;
106     $studentColor = function ($id) use ($steward_id, $student_id) {
107         if ($id == $steward_id) {
108             return 'success';
109         }
110         if ($id == $student_id) {
111             return 'warning';
112         }
113         return '';
114     };
115
116     return view('group.show', compact('group', 'studentColor', 'canEdit'));
117 }
118
119 public function edit(Group $group)
120 {
121     $this->restrictSteward($group);
122     return view('group.edit', compact('group'));
123 }
124
125 private function restrictSteward(Group $group)
126 {
127     if ($group->steward_id != auth()->user()->student_id) {

```



```

128         abort(403, 'Для того, что бы зайти на эту страницу необходимо быть старостой этой группы');
129     }
130 }
131
132 public function delete(Group $group)
133 {
134     $this->restrictSteward($group);
135     return view('group.delete', compact('group'));
136 }
137
138 public function stat(Group $group)
139 {
140     $weeks = [];
141     $stats = [];
142
143     $students = $group->students()->get(['id', 'name']);
144
145     $student_ids = array_unique(array_map(function ($el) {
146         return $el->id;
147     }, $students->all()));
148     $attendances = Attendance::whereIn('student_id', $student_ids)->get(['class_id', 'student_id',
149 'presence', 'reason']);
150
151     $class_ids = array_unique(array_map(function ($el) {
152         return $el->class_id;
153     }, $attendances->all()));
154     $classes = [];
155     foreach (Clazz::whereIn('id', $class_ids)->get(['id', 'date'])->all() as $class) {
156         $classes[$class->id] = $class;
157     }
158
159     foreach ($student_ids as $student_id) {
160         $stats[$student_id] = [
161             'weeks' => [],
162             'total' => [
163                 'no_reason' => 0,
164                 'total' => 0,
165             ],
166             'pos' => -1,
167         ];
168     }
169     foreach ($attendances as $atd) {
170         $class = $classes[$atd->class_id];
171
172         $week_start = $class->date->copy()->startOfWeek();
173         $week_end = $class->date->copy()->endOfWeek();
174         $weed_id = $week_start->format('Y/m/d');
175         if (!array_key_exists($weed_id, $weeks)) {
176             $weeks[$weed_id] = [
177                 'start' => $week_start,
178                 'end' => $week_end,
179             ];
180             foreach ($student_ids as $student_id) {
181                 $stats[$student_id]['weeks'][$weed_id] = [
182                     'no_reason' => 0,
183                     'total' => 0,
184                 ];
185             }
186         }
187         $student_id = $atd->student_id;
188
189         if (!array_key_exists($weed_id, $stats[$student_id]['weeks'])) {
190             $stats[$student_id]['weeks'][$weed_id] = [
191                 'no_reason' => 0,
192                 'total' => 0,
193             ];
194         }
195
196         $total = &$stats[$student_id]['total'];
197         $week = &$stats[$student_id]['weeks'][$weed_id];
198         if (!$atd->presence) {
199             $week['total'] += 1;
200             $total['total'] += 1;
201             if ($atd->reason == null) {
202                 $week['no_reason'] += 1;
203                 $total['no_reason'] += 1;
204             }
205         }
206     }
207     ksort($weeks);
208
209     $k = 1;

```

```

210 while ($k <= count($stats)) {
211     foreach ($stats as $i => $i_stat) {
212         if ($stats[$i]['pos'] != -1) {
213             continue;
214         }
215         $max = $i;
216         foreach ($stats as $j => $j_stat) {
217             if ($j_stat['pos'] != -1) {
218                 continue;
219             }
220             if ($stats[$max]['total']['total'] < $j_stat['total']['total']) {
221                 $max = $j;
222             }
223         }
224         $stats[$max]['pos'] = $k++;
225     }
226 }
227
228 return view('group.stat', compact('group', 'students', 'weeks', 'stats'));
229 }
230
231 public function list(Group $group)
232 {
233     $super = null;
234     $available = [];
235     if ($group->super) {
236         $super = $group->super;
237         $available = $super->students->all();
238     }
239     $list = array_map(function ($el) {
240         return $el->id;
241     }, $group->students->all());
242     $isSelected = function ($id) use ($list) {
243         return in_array($id, $list);
244     };
245
246     return view('group.list', compact('group', 'available', 'isSelected'));
247 }
248
249 public function list_update(Request $request, Group $group)
250 {
251     $this->restrictSteward($group);
252     $available = [];
253     if ($group->super) {
254         $available = $group->super->students->all();
255     }
256     $selected = [];
257     foreach ($available as $el) {
258         if ($request->has('student-' . $el->id) && $request->get('student-' . $el->id)) {
259             $selected[] = $el->id;
260         }
261     }
262     $group->students()->sync($selected);
263
264     return redirect()->route('group.show', [$group]);
265 }
266
267 public function store(Request $request)
268 {
269     /* @var Student $student */
270     $student = auth()->user()->student;
271
272     $this->validate($request, [
273         'name' => 'required|string',
274         'course' => 'required|integer|min:1|max:9',
275         'super_id' => [
276             'required',
277             'integer',
278             Rule::exists('group', 'id')->where('steward_id', $student->id),
279         ],
280     ]);
281
282     $cathedral_id = Group::find($request->get('super_id'), ['cathedral_id']->cathedral_id;
283     $student = auth()->user()->student;
284     $group = new Group($request->all());
285     $group->cathedral_id = $cathedral_id;
286     $group->super_id = $request->super_id;
287     $student->stewarded()->save($group);
288
289     return redirect()->route('group.index');
290 }
291
292 public function update(Request $request, Group $group)

```

```

293     {
294         $this->restrictSteward($group);
295
296         $this->validate($request, [
297             'name' => 'required|string',
298             'course' => 'required|integer|min:1|max:9'
299         ]);
300
301         $group
302             ->fill($request->all())
303             ->save();
304
305         return redirect()->route('group.index');
306     }
307
308     public function destroy(Group $group)
309     {
310         $this->restrictSteward($group);
311
312         $group->delete();
313
314         return redirect()->route('group.index');
315     }
316 }

```

Текст контроллера занятий:

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Attendance;
6  use App\Clazz;
7  use App\Discipline;
8  use App\Group;
9  use App\Student;
10 use App\Teacher;
11 use Carbon\Carbon;
12 use Illuminate\Http\Request;
13 use Illuminate\Validation\Rule;
14
15 class ClassController extends Controller
16 {
17     /**
18      * Create a new controller instance.
19      */
20     public function __construct()
21     {
22         $this->middleware(['auth', 'student']);
23         $this->middleware(['steward', ['only' => ['create', 'store', 'edit', 'delete', 'update',
24 'destroy']]));
25     }
26
27     public function index(Request $request)
28     {
29         /* @var Student $student */
30         $student = auth()->user()->student;
31
32         $date = Carbon::now();
33         if ($request->has('date')) {
34             $date = Carbon::parse($request->get('date'));
35         }
36
37         $today = $date->copy()->startOfDay();
38         $yesterday = $today->copy()->subDay();
39         $tomorrow = $today->copy()->addDay();
40
41         $groups = array_map(function ($el) {
42             return $el->id;
43         }, $student->groups()->get(['id'])->all());
44         foreach (Group::where('steward_id', $student->id)->get(['id'])->all() as $el) {
45             $groups[] = $el->id;
46         }
47         array_unique($groups);
48         $classes = Clazz::whereIn('group_id', $groups)
49             ->whereBetween('date', [$today, $today->copy()->endOfDay()])
50             ->with('discipline', 'teacher')
51             ->ordered()
52             ->get()
53             ->all();

```

```

53
54     $classStatus = function (Clazz $class) {
55         $now = Carbon::now();
56         if ($now->lt($class->date)) {
57             return 'future';
58         }
59         if ($now->lte($class->duration)) {
60             return 'present';
61         }
62         return 'past';
63     };
64
65     return view('class.index', compact('classes', 'today', 'yesterday', 'tomorrow',
66 'classStatus'));
67 }
68
69 public function create()
70 {
71     /* @var Student $student */
72     $student = auth()->user()->student;
73
74     $class = new Clazz();
75     $disciplines = [];
76     foreach (Discipline::all(['id', 'name']) as $el) {
77         $disciplines[$el->id] = $el->name;
78     }
79     asort($disciplines);
80     $teachers = [];
81     foreach (Teacher::all(['id', 'name']) as $el) {
82         $teachers[$el->id] = $el->name;
83     }
84     asort($teachers);
85     $groups = [];
86     foreach ($student->stewarded()->getQuery()->get(['id', 'name'])->all() as $el) {
87         $groups[$el->id] = $el->name;
88     }
89     $types = [
90         'lection' => 'Лекция',
91         'laboratory' => 'Лабораторная работа',
92         'practice' => 'Практическое занятие',
93         'seminar' => 'Семинар',
94         'course' => 'Курсовая работа',
95     ];
96     $repeats = [
97         null => 'Нет',
98         'once' => 'Раз в 2 недели',
99         'twice' => 'Каждую неделю',
100     ];
101     return view('class.create', compact('class', 'disciplines', 'teachers', 'groups', 'types',
102 'repeats'));
103 }
104
105 public function edit(Clazz $class)
106 {
107     /* @var Student $student */
108     $student = auth()->user()->student;
109     $disciplines = [];
110     foreach (Discipline::all(['id', 'name']) as $el) {
111         $disciplines[$el->id] = $el->name;
112     }
113     asort($disciplines);
114     $teachers = [];
115     foreach (Teacher::all(['id', 'name']) as $el) {
116         $teachers[$el->id] = $el->name;
117     }
118     asort($teachers);
119     $groups = [];
120     foreach ($student->stewarded()->getQuery()->get(['id', 'name'])->all() as $el) {
121         $groups[$el->id] = $el->name;
122     }
123     $types = [
124         'lection' => 'Лекция',
125         'laboratory' => 'Лабораторная работа',
126         'practice' => 'Практическое занятие',
127         'seminar' => 'Семинар',
128         'course' => 'Курсовая работа',
129     ];
130     return view('class.edit', compact('class', 'disciplines', 'teachers', 'groups', 'types'));
131 }
132
133 public function update(Request $request, Clazz $class)

```

```

134 {
135   /* @var Student $student */
136   $student = auth()->user()->student;
137
138   $this->validate($request, [
139     'group_id' => [
140       'required',
141       'integer',
142       Rule::exists('group', 'id')->where('steward_id', $student->id),
143     ],
144     'discipline_id' => 'required|integer|exists:discipline,id',
145     'teacher_id' => 'integer|exists:teacher,id',
146     'date' => [
147       'required',
148       'date_format:Y-m-i\TH:i'
149     ],
150     'duration' => [
151       'required',
152       'date_format:H:i'
153     ],
154     'location' => 'required',
155     'type' => 'required|in:lection,laboratory,practice,seminar,course',
156   ]);
157
158   $class->fill($request->all());
159   $class->group_id = $request->get('group_id');
160   $class->discipline_id = $request->get('discipline_id');
161   $class->teacher_id = $request->get('teacher_id');
162   $class->save();
163
164   return redirect()->route('class.show', $class);
165 }
166
167 public function store(Request $request)
168 {
169   /* @var Student $student */
170   $student = auth()->user()->student;
171
172   $this->validate($request, [
173     'group_id' => [
174       'required',
175       'integer',
176       Rule::exists('group', 'id')->where('steward_id', $student->id),
177     ],
178     'discipline_id' => 'required|integer|exists:discipline,id',
179     'teacher_id' => 'integer|exists:teacher,id',
180     'date' => [
181       'required',
182       'date_format:Y-m-i\TH:i'
183     ],
184     'duration' => [
185       'required',
186       'date_format:H:i:s'
187     ],
188     'location' => 'required',
189     'type' => 'required|in:lection,laboratory,practice,seminar,course',
190     'repeat' => 'sometimes|in:once,twice',
191     'repeat_till' => 'sometimes|date_format:Y-m-d|after:now',
192   ]);
193
194   $repeat = 0;
195   switch ($request->get('repeat', 'none')) {
196     case 'once':
197       $repeat = 2;
198       break;
199     case 'twice':
200       $repeat = 1;
201       break;
202   }
203   $repeat_till = Carbon::parse($request->get('repeat_till'));
204
205   $class = new Clazz($request->all());
206   $class->group_id = $request->get('group_id');
207   $class->discipline_id = $request->get('discipline_id');
208   $class->teacher_id = $request->get('teacher_id');
209   $class->save();
210   if ($repeat) {
211     /* @var Carbon $date */
212     $date = $class->date;
213     while (($date = $date->copy()->addWeek($repeat))->lte($repeat_till)) {
214       /* @var Clazz $class */
215       $class = $class->replicate();
216       $class->date = $date;

```

```

217         $class->save();
218     }
219 }
220 return redirect()->route('class.index');
221 }
222
223 public function delete(Clazz $class)
224 {
225     $this->restrictSteward($class);
226     return view('class.delete', compact('class'));
227 }
228
229 private function restrictSteward(Clazz $class)
230 {
231     if ($class->group()->getQuery()->first(['steward_id'])->steward_id != auth()->user()-
>student_id) {
232         abort(403, 'Для того, что бы зайти на эту страницу необходимо быть старостой группы');
233     }
234 }
235
236 public function destroy(Request $request, Clazz $class)
237 {
238     $this->restrictSteward($class);
239     $class->delete();
240     return redirect()->route('class.index');
241 }
242
243 public function show(Clazz $class)
244 {
245     /* @var Student $student */
246     $student = auth()->user()->student;
247
248     $students = [];
249     foreach ($class->students as $stud) {
250         $students[$stud->id] = $stud->name;
251     }
252     foreach ($class->attendances as $attendance) {
253         $students[$attendance->student_id] = $attendance;
254     }
255     $reasons = [
256         'ill' => 'Болезнь',
257         'conference' => 'Конференция',
258         'other' => 'Другое',
259     ];
260     $canEdit = $class->group->steward_id == $student->id;
261     return view('class.show', compact('class', 'canEdit', 'students', 'reasons'));
262 }
263
264 public function attendance(Request $request, Clazz $class)
265 {
266     $this->restrictSteward($class);
267     $this->validate($request, [
268         'student.presence' => 'sometimes|boolean',
269         'student.reason' => 'in:ill,conference,other',
270         'student.mark' => 'sometimes|integer|min:0|max:100',
271     ]);
272
273     foreach ($request->get('student', []) as $stud_id => $stud) {
274         /* @var Attendance $attendance */
275         $attendance = Attendance::firstOrCreate(['class_id' => $class->id, 'student_id' => $stud_id]);
276         if (!$attendance->id) {
277             $attendance->class_id = $class->id;
278             $attendance->student_id = $stud_id;
279         }
280         $attendance->presence = (isset($stud['presence']) ? $stud['presence'] : false) ? 1 : 0;
281         if ($attendance->presence) {
282             $attendance->reason = null;
283         } else {
284             $attendance->reason = (isset($stud['reason']) && !empty($stud['reason'])) ?
$stud['reason'] : null;
285         }
286         $attendance->mark = (isset($stud['mark']) && !empty($stud['mark'])) ? $stud['mark'] : null;
287
288         $attendance->save();
289     }
290     return redirect()->route('class.show', [$class]);
291 }
292 }

```