ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА ДЛЯ МУЗЫКАЛЬНОЙ ШКОЛЫ

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии
очной формы обучения, группы 07001401
Козырева Егора Алексеевича

Научный руководитель
к.т.н., доцент Бурданова Е.В.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача создания автоматизированных систем учета не является редкостью в наше время. Создание такой системы, во-первых – существенно упростит работу предприятия, автоматизируя большую часть объемной работы, которая возлагается на персонал, а во-вторых, обеспечит клиенту постоянный доступ к личным данным, а также ряд действий, производимых в рамках собственной учетной записи, таких как: распоряжение балансом, изменение личных данных и прочее.

Традиционно для решения этой задачи используют различные готовые онлайн сервисы. Однако они обладают лишь базовым набором часто-используемого функционала, далеко не всегда удовлетворяют требованиям различных учреждений, имеющих некие особенности или различия в организации рабочего процесса. К тому же они являются платными и имеют дополнительный ряд ограничений, связанных с тарифным планом и лишены возможности в доработке.

Целью данной работы является разработка и реализация автоматизированной системы учета для частной музыкальной школы, имеющей ряд нестандартных требований к реализации рабочего процесса учреждения.

Для достижения поставленной цели необходимо решить и реализовать ряд поставленных задач:

– реализация личных кабинетов пользователей (преподавателя и клиента);

– возможность просмотра и контроля над личными кабинетами пользователей для менеджеров и администраторов;

– баланс, где у пользователя будет возможность распоряжаться своими финансами, а именно – покупать интересующие тарифы (уроки или набор уроков за определенную цену);

– история денежных операций клиентов и преподавателей для формирования финансовой отчетности;

– организовать работу с лидами (потенциальными клиентами);

– формирование статистики по различным критериям за указанный период;

– расписание занятий; менеджерам будут видны все занятия, возможность редактирования графика, добавление и удаление занятий, а преподавателям и клиентам доступны лишь занятия, в которых они задействованы;

– панель администратора для полного контроля данных в системе, не имеющая ограничений в плане добавления, редактирования или удаление каких-либо записей;

Актуальность данной работы заключается в потребности музыкальной школы автоматизации процессов и создании удобного интерфейса для доступа к необходимым данным пользователю в любое время.

Уникальность разработанной системы заключается в выбранном подходе формирования расписания и описана в п. 1.3. «Формирование требований к системе».

В первой главе «Анализ предметной области и постановка задачи» производится обзор уже существующих систем, применяемых для подобного рода задач, детальное изучение предметной области, конкретных требований заказчика, а также обоснование предложенных методов и алгоритмов, для достижения поставленной цели.

Во второй главе «Проектирование системы» будут описаны основные принципы работы системы, взаимодействия разных её частей и проектирование базы данных.

В третьей главе «Программная реализация» будет произведена разработка структуры сайта и выбор оптимального инструмента для разработки и программной реализации.

Работа содержит 43 страницы, 21 рисунок, 1 таблица и 2 приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

1.1. Изучение особенностей рабочего процесса музыкальной школы

Структура школы «Musicmetod» не имеет в своем составе каких-либо самостоятельных отделов или департаментов, разве что имеет два филиала по г. Белгород. Рабочий процесс этих двух филиалов очень схож. Они оперируют одной и той же базой данных клиентов, тарифов, групп и т.д. Различие заключается лишь в количестве классов для проведений занятий.

В работе школы было выделено восемь основных пунктов, некоторые из которых связанны между собой, а некоторые нет:

– пользователи:
    – администраторы;
    – менеджеры;
    – преподаватели;
    – клиенты;
– группы (групповые занятия);
– филиалы;
– расписание:
    – текущее;
    – основное;
– финансы:
    – зачисление;
    – списание;
    – выплаты;
– лиды;
– статистика;
– задачи.

Главным в иерархии пользователей идет администратор, он же директор. Имеет неограниченный доступ к любой информации и возможности касательно всей системы. Основной задачей директора является руководство и организация всего рабочего процесса школы.

Следующие в иерархии идут менеджеры. Обычно один менеджер курирует работу одного филиала и не имеют столько прав и возможностей, какие есть у директора. К примеру, менеджер может создать пользователю платеж, добавлять некоторое количество комментариев к платежу, однако редактировать сумму платежа или комментарии, а также удалять что-либо может лишь директор, то есть администратор.

Далее идут клиенты и преподаватели. Клиенты имеют право распоряжаться своим балансом – покупать различные тарифы на индивидуальные или групповые занятия, однако самостоятельно пополнить баланс они не могут. Из личного кабинета клиента ему ещё доступна функция смены логина или пароля и доступ к информации только о расписании своих занятий. Так же клиент может состоять в одной или нескольких групп. Для сотрудника, в свою очередь, доступна лишь информация о своем расписании.

Группы пользователей созданы для возможности реализации групповых занятий. У группы есть название, учитель и список учеников, принадлежащих данной группе. Один ученик также может принадлежать нескольким группам одновременно.

Филиалы представляют собой помещение для проведения занятий. Содержит название и количество классов (на данный момент у школы есть 2 филиала, названных по названию улицы, на которой они расположены - «Мичурина» и «Щорса», которые размещают в себе 5 и 2 класса соответственно). Как видно на рисунке №1 с филиалами также связаны занятия в расписании.

Расписание является одной из самых важных и в то же время сложных частей системы. Основной график – список занятий для филиалов на каждый

день недели. Основной график дублируется, но с учетом на даты добавления или удаления занятий из расписания. Текущий график представляет собой дублирование основного графика с учетом различных обстоятельств, которые будут описаны в пункте 1.3. «Формирование требований к системе», и разовые занятия, которые актуальны лишь на один день и не будут дублироваться вместе с основным графиком.

В свою очередь занятие принадлежит одному из филиалов и содержит в себе информацию о преподавателе, ученике или группе учеников, время начала и конца урока.

Раздел «Финансы» содержит в себе информацию о всех денежных операциях. Любая денежная операция содержит в себе дату, сумму, тип операции (зачисление, списание и т.д.) и может содержать один или несколько комментариев.

Статистика доступна лишь директору. Отвечает за формирование отчетов за выбранный период по различным критериям.

Раздел «Лиды» посвящен людям, которые только связались со школой и обсуждают интересующие их вопросы, связанные с обучением. В конечном счете кто-то из них становится клиентом, а кто-то отказывается.

Задачи выставляются любым менеджером или преподавателем и доступны менеджерам или директору. Обычно это различного рода примечания или форма связи. К примеру, один из менеджеров договорился созвониться с клиентом через 2 дня и, дабы поставить в известность другого менеджера, который будет в тот день об этом – просто создается задача с текстом: «позвонить Ивану Ивановичу по телефону 555-555 11.04 после обеда». Тот менеджер, который созвонился с клиентом закрывает эту задачу.

1.2. Обзор существующих сервисов для решения подобных задач

В качестве первого примера будет рассмотрена CRM система «Listok» [24]. Данная система имеет жесткие ограничения функционала в зависимости от тарифа. К примеру самый дешевый тариф стоит 1000 руб./мес. и содержит в себе лишь расписание занятий, рассылки и отчеты. Второй тариф, стоимостью 1600 руб./мес. отличия от первого тарифа – это наличие личного кабинета и возможность создания до двух филиалов.

Даже если покупать более дорогой тариф – всё равно функционал сильно ограничен. Однако, не взирая на тарифный план, предлагаемая система обладает рядом существенных недостатков, связанных с расписанием:

– дело в том, что данное расписание не дает возможности выставлять разовые занятия, а также выставлять период отсутствия ученика так, чтобы образовывалось окно в графике, но менеджер понимал, что это время свободно лишь на определенный период и не ставил в этот промежуток каких-либо других занятий;

– невозможность разовой корректировки времени занятия. К примеру: клиент связался с менеджерами и предупредил о том, что не может прийти сегодня в назначенное ему время и просит переставить ему время занятий лишь на один день так, чтобы это никак не повлияло на дальнейшее расписание;

– отсутствие возможности выставления в расписание индивидуального урока.

В качестве второго примера была рассмотрена CRM система «Hollihop» [25]. Эта система, в отличии от предыдущей, обладает большим функционалом и предлагает 3 тарифных плана. Для сравнения с собственной системой будет взят самый дорогой тариф стоимостью 7000 руб./мес. Сравнение, для наглядности, будет показано в таблице 1.1. В данной системе расписание предлагает возможность разделения занятий на индивидуальные

и групповые, но обладает таким же рядом недостатков, которые были описаны выше для предыдущей системы, что, в свою очередь, не дает возможности максимально рационального использования времени и «гибкости».

К тому же ни одну из предоставленных систем нельзя будет доработать под нетривиальные задачи или особые пожелания клиента, поскольку отсутствует доступ к файлам проекта, что является большим минусом.

Опираясь на то, что расписание является самой приоритетной частью системы (по словам заказчика) и обе эти системы подходили под требования лишь частично – было принято решение не использовать готовые решения и создать максимально гибкую и функциональную систему, обладающую достоинствами предлагаемых CRM систем и с доработкой различных их частей под конкретные потребности данной школы.

Таблица 1.1

Сравнение готового решения с разработанной системой

	«Hollihop»	Разработанная система
Ученики (активные)	До 400	Неограниченно
Расписание	Да	Да
База данных учеников	Да	Да
Отчеты и статистика	Да	Да
История коммуникаций с клиентами	Да	Да/Нет
Финансы	Да	Да
Блок задач для сотрудников	Да	Да
Личный кабинет	Дополнительно от 1000 руб/мес	Да
Доступ к выгрузке данных в MS Excel	Да	Нет

Из таблицы 1 можно заметить, что готовое решение имеет ограничение по количеству активных клиентов, а также имеет возможность выгрузки

данных в MS Excel. Ограничение по количеству активных пользователей убирается лишь при покупке индивидуального плана, что будет стоить ещё дороже.

В свою очередь разработанная система в рамках ВКР не имеет ограничения по количеству активных учеников и не требует доплаты за личный кабинет, но в тоже время отсутствует возможность выгрузки данных в MS Excel и история коммуникаций реализована лишь с «лидами», обычные клиенты имеют примечания, которые видны и редактируются только менеджерами или администраторами.

1.3. Формирование требований к системе

Поскольку список требований довольно велик – в данном пункте будут описаны лишь самые важные из них без лишних подробностей.

Первый, а также один из самых важных моментов – разделение прав доступа к различным частям системы, так как это обеспечивает безопасность и достоверность имеющихся данных. Всего в данном проекте выделено 4 группы пользователей (администратор, менеджер, учитель и клиент) и у каждой из них имеются свои особенности и ограничения:

– директор имеет доступ к абсолютно любой части системы с возможностью полного контроля данных (добавление, удаление, редактирование);

– директор и менеджер имеют возможность просматривать личные кабинеты преподавателей и клиентов;

– покупка тарифов клиенту может производиться как самим клиентом, так и менеджером или директором;

– клиентам и преподавателям доступно лишь свое расписание, то есть они не могут видеть расписание других занятий, которые с ими не связаны;

– менеджерам и директору доступна полная картина расписания всех учителей и учеников;

– блок с примечаниями и последней датой авторизации клиента виден лишь менеджерам или директору;

– возможность добавления комментариев к платежам доступна лишь менеджерам и директору, однако редактирование содержимого комментария или его удаление доступно лишь директору;

– разделы: «задачи», «лиды», «пользователи» (клиенты, штат, архив), «группы» доступны менеджерам и директору;

– менеджерам недоступны некоторые разделы, такие как «тарифы» «Финансы» и «статистика»;

– клиентам доступны разделы: «баланс», «расписание» и «смена логина/пароля»;

– учителям доступен лишь раздел «расписание».

Личные кабинеты доступны ученикам и учителям. У менеджеров и директора личных кабинетов нет. Формат личного кабинета имеет некие различия в зависимости от того просматривается ли он от лица клиента или менеджера.

Личный кабинет клиента от лица менеджера:

– раздел «баланс»;

– раздел «расписание».

Личный кабинет клиента от лица клиента:

– раздел «баланс»;

– раздел «расписание»;

– раздел «сменить логин или пароль».

Личный кабинет преподавателя от лица менеджера:

– навигационное меню:

– раздел «расписание»;

– таблица с отправкой данных о проведенных занятиях;

– таблица выплат.

Личный кабинет преподавателя от лица преподавателя:

– навигационное меню;

– раздел «расписание»;

– таблица с отправкой данных о проведенных занятиях;

– кнопка «написать администратору».

Раздел «Финансы» состоит из таблицы платежей. Платеж содержит в себе сумму, дату, принадлежность какому-либо пользователю, тип (зачисление, списание и т.д.) и комментарии (примечания).

Раздел «Пользователи» разделен на три подраздела: «Штат», «Клиенты» и «Архив». Раздел «Клиенты» представляет из себя таблицу пользователей, принадлежащих группе «Клиенты». В таблице имеется краткая информация о пользователе и есть возможность отправить учетную запись в архив, зачислить платеж или редактировать данные о пользователе. Так же в этом разделе присутствует функционал для создания нового пользователя.

Раздел «Лиды» представляет из себя таблицу с данными о потенциальных клиентах и комментариями менеджеров, описывающие текущее положение дел с тем или иным человеком.

Раздел «Группы» содержит в себе таблицу с информацией о всех имеющихся группах, а также возможность добавления, редактирования и удаления.

Задачи представляют из себя примечания либо обсуждение какой-либо проблемы. Задачи могут быть двух видов:

– «до выполнения» - задача активна пока её не отметят как завершенную;

– «на сегодня» – задачи появляются в списке лишь в определенный день.

Сам раздел делится на 3 подраздела:

– общий список задач - список абсолютно всех задач;

– список задач «на сегодня» - список задач типа «на сегодня» начиная с текущей даты и далее;

– актуальные задачи – список задач типа «на сегодня» и имеющие текущую дату, а также все активные задачи типа «до выполнения».

Раздел «Статистика» формирует отчеты по разным критериям за определенный период. Список критериев:

– вывод информации по лидам;

– данные по проведенным урокам;

– данные по зарплатам преподавателей;

– статистика по клиентам и их балансам (без учета периода).

Раздел тарифов содержи в себе информацию о всех имеющихся тарифах и набор инструментов для полного контроля над данными. Тариф представляет собой количество индивидуальных или групповых занятий за определенную цену. Данный раздел доступен лишь администратору (директору).

Раздел «Расписание» представляет собой таблицу со списком занятий на определенный день. Расписание должно быть разделено на «основное» и «текущее». Именно этот подход делает систему в своем роде уникальной, так как ни одна из ныне существующих систем не может предоставить подобный функционал. Особенности и преимущества такого подхода описаны ниже.

Занятия, принадлежащие основному расписанию, дублируются в определенный день недели и время, начиная с даты добавления и до даты удаления из графика.

Занятия, принадлежащие текущему расписанию, по большей степени, дублируют основной график. Список причин, по которым занятие не будет продублировано из основного графика:

– дата проведения занятия совпадает с периодом отсутствия ученика;

– у занятия в этот день стоит пометка о разовом отсутствии ученика на занятии.

Так же занятия основного графика могут дублироваться в текущий с изменением по времени проведения, если менеджер внес данные изменения.

У текущего графика есть ещё одна важная функция – это добавление в расписание разового занятия только на конкретную дату и время без последующего его дублирования. К примеру, если новый клиент ещё не определился с графиком занятий, то рано добавлять его в основное расписание, можно поставить ему занятие в текущий график в любое свободное время и эта запись не будет дублироваться на следующие недели и делать так каждый раз до тех пор, пока не будет утвержден точный график, после чего занятия клиента можно добавить в основное расписание.

Еще одним плюсом такого подхода является то, что если клиент временно не может посещать занятия (выставлен период отсутствия) в текущем графике его занятия появляться не будут, однако в основном графике будет видно, что это время свободно лишь временно, дабы менеджер не добавил занятия в основной график в это время другому клиенту и мог ставить только разовые занятия в текущий график в образовавшееся временное окно.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Разработка структуры системы

В качестве схемы распределения данных был выбран шаблон проектирования MVC. Схема его работы представлена на рисунке 2.

MVC (Model-View-Controller) – «Модель-Представление-Контроллер» это схема распределения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента таким образом, что модификация каждого компонента может осуществляться независимо [17].

Модель представляет данные и реагирует на команды контроллера, изменяя свое состояние.

Представление отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основная цель применения этой концепции стоит в отделении бизнес-логики от её визуализации. За счет такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности выполняются следующие задачи:

– к одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели;

– не затрагивая реализацию видов, можно изменить реакции на действия пользователя – для этого достаточно использовать другой контроллер;

– ряд разработчиков специализируется только в одной из областей: либо разрабатывает графический интерфейс, либо разрабатывает бизнес-логику;

Рис. 2.1. Схема работы шаблона проектирования MVC

Следующая используемая технология в проекте – ORM (объективно реляционное отображение, или преобразование). Это технология программирования, связывающая базы данных и объектно-ориентированные языки программирования, создавая «виртуальную объектную базу данных» [6].

Общая структура разделов и доступов к ним различных групп пользователей представлена на рисунке 2.1. Каждый раздел является реализацией одной из подзадач и предоставляет интерфейс для работы с её компонентами.

Доступ к какому-либо разделу требует авторизации в системе. Во время формирования страницы проверяются права доступа авторизованного пользователя к разделу. Это является очень важным требованием для обеспечения целостности и достоверности данных.

Каждый раздел состоит из горизонтального навигационного меню и контентной части страницы.

Самым важным и сложным был раздел расписания, так как является нетривиальной задачей, не имеющей аналогов и объединяет в себе сразу несколько частей системы. Структура формирования расписания показана на рисунке 2.2.

Все остальные разделы имеют довольно простую структуру и либо слабо связаны между собой, либо являются самостоятельными. Каждый раздел, по большей степени, состоит из таблицы и пользовательского интерфейса для управлением данными раздела.

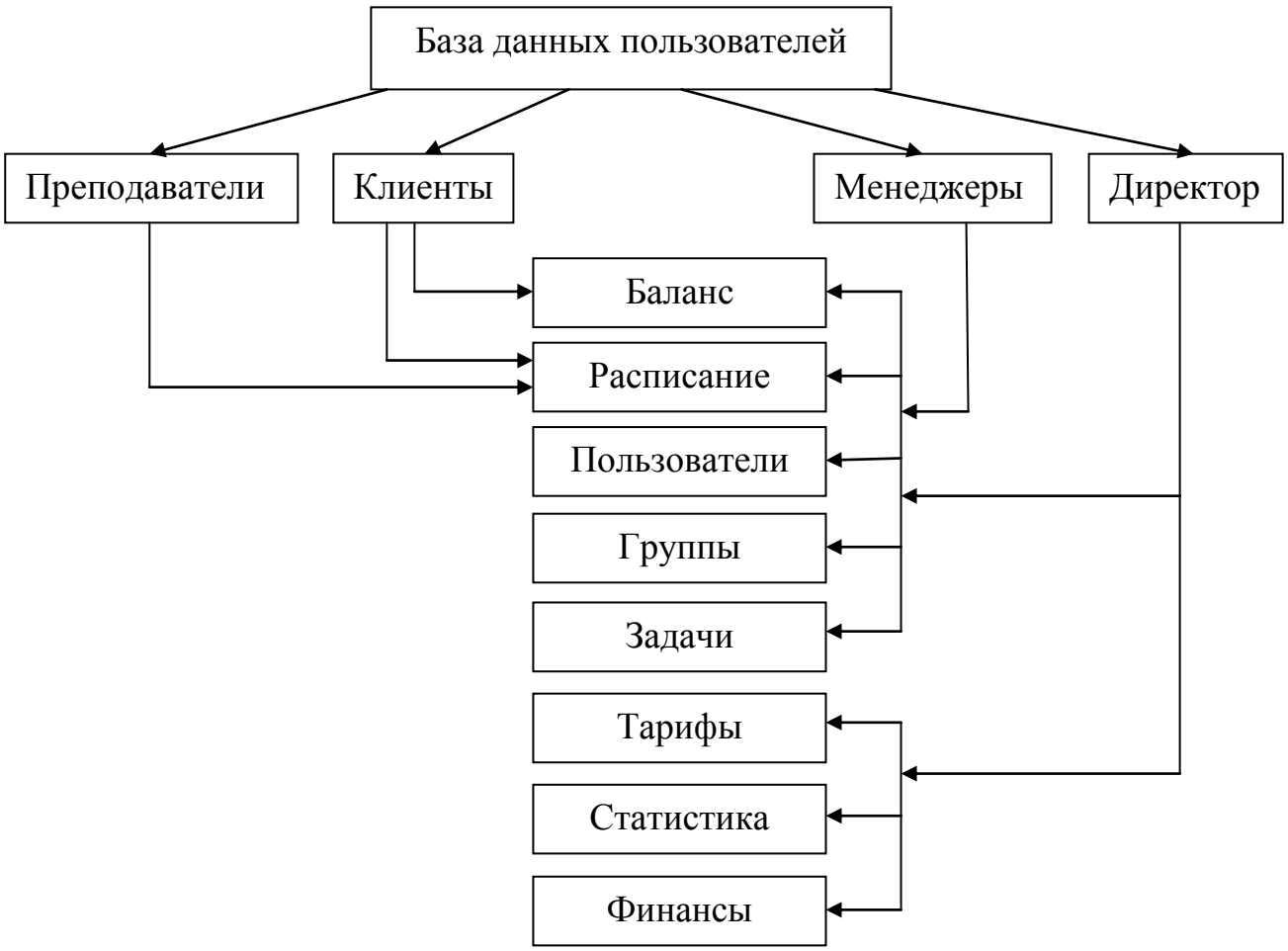


Рис. 2.2. Общая структура разделов

В свою очередь внешний вид и функционал страниц может иметь незначительные отличия в зависимости от прав доступа авторизованного пользователя. К примеру, в личном кабинете ученика только менеджеру доступны примечания к клиенту, дата и время его последней авторизации, интерфейс для зачисления денежных средств на баланс клиента и добавление комментариев к платежам.

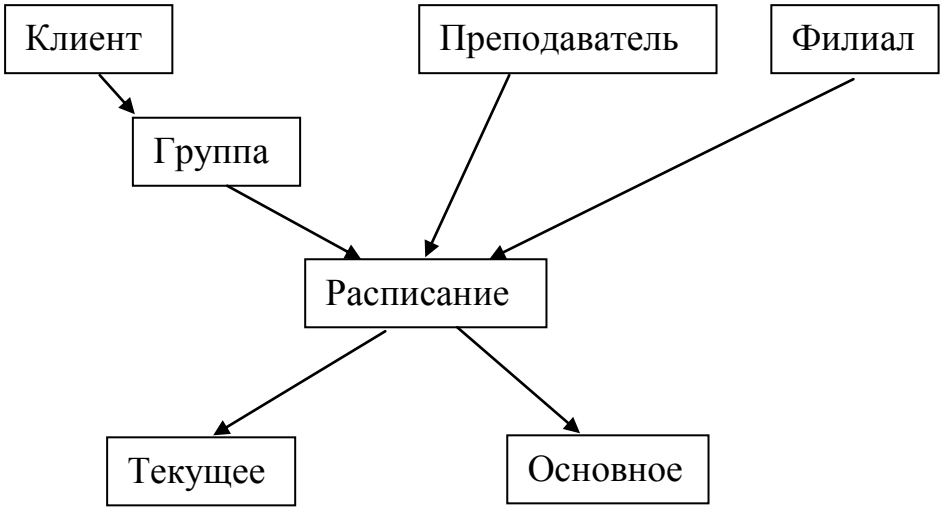


Рис. 2.3. Общая структура расписания

2.2. Проектирование структуры базы данных

Данная система работает с базой данных MySQL. Эта база данных является одной из самых используемых. Выбор пал на MySQL по большей степени из-за того, что её по умолчанию поддерживает практически любой хостинг. Другие преимущества данной базы данных описаны ниже [18].

Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко.

Довольно высокий уровень безопасности обеспечивается благодаря базе данных MySQL, создающейся при установке пакета и содержащей пять таблиц. При помощи этих таблиц можно описать, какой пользователь из какого домена с какой таблицей может работать и какие команды он может применять.

Раньше лицензирование было немного запутанным, но сейчас для некоммерческих целей данная БД распространяется бесплатно.

Структура базы данных была приведена к третьей нормальной форме.

Нормальная форма – требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных

функциональных зависимостей между атрибутами (полями таблиц). Отношение находится в третьей нормальной форме, когда находится во второй и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы [15].

Входе логического проектирования было выделено несколько основных сущностей: пользователь, группа пользователя, задача, примечание к задаче, тип задачи, платеж, тип платежа, группа клиентов, занятие, тип занятия, филиал. Инфологическая модель базы данных показана на рисунке 2.4. Логическая модель базы данных показана на рисунке 2.5. [11]

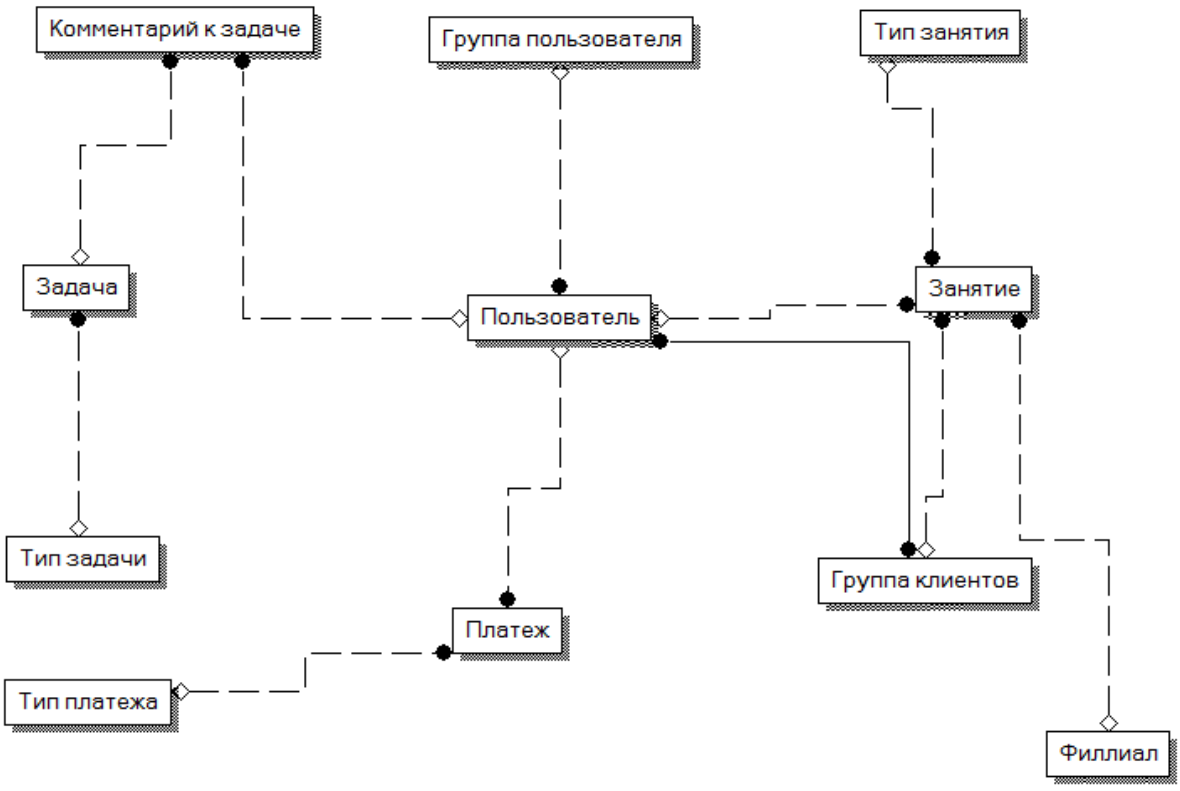


Рис. 2.4. Инфологическая модель базы данных

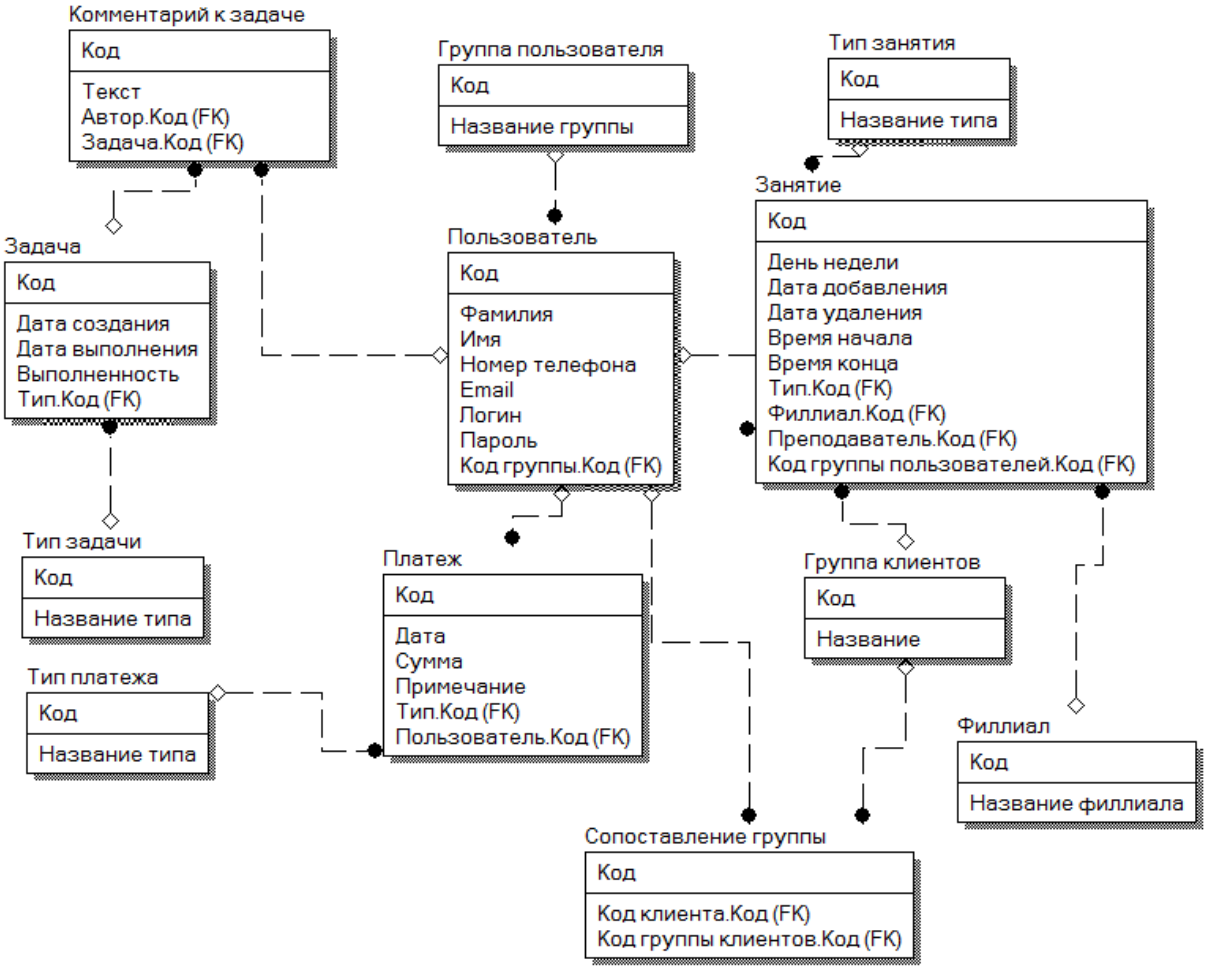


Рис. 2.5. Логическая модель базы данных

2.3. Выбор программных средств

Для разработки системы был выбран язык программирования PHP за его практичность и гибкость. Существует ещё одна «характеристика», которая делает этот язык особенно привлекательным: он распространяется бесплатно, причем с открытым исходным кодом. Далее будут расписаны другие особенности данного языка [16].

В качестве среды разработки использовался «PhpStorm», поскольку большие проекты проще и удобнее создавать при помощи полноценный IDE (интегрированный средств программирования).

PhpStorm как IDE отличается от текстовых редакторов [23]:

– индексирует файлы проекта, чтобы потом мгновенно выполнять по ним поиск;
– распознавание контекста.

Также данная среда разработки обладает рядом очень полезных инструментов:

– работа с сервисами контроля версий «Git» и «GitHub»;
– доступ к базе данных позволяет выполнять запросы из среды разработки;
– доступ к серверу по SSH/FTP для загрузки файлов проекта.

Также в проекте был использован язык программирования Javascript для упрощения пользовательского интерфейса и минимизации нагрузки на сервер, а именно популярная библиотека – Jquery [3].

Исходя из пожеланий заказчика было принято решение об использовании технологии AJAX (Asynchronous Javascript And Xml) –подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с сервером. В результате, при обновлении данных веб-страница не перезагружается полностью и веб-приложение становится быстрее и удобнее [1].

Для формирования HTML кода был выбран XSLT-шаблонизатор. В общем случае, XSLT-шаблон представляет собой XML-документ, описывающий правила преобразования исходных данных XML-документов. Важной особенностью является то, что с помощью XSLT-шаблонов можно получать различные представления одного и того же XSL-ресурс. Схема работы страницы с применением простого XSLT-шаблона изображена на рисунке 2.6. [14]

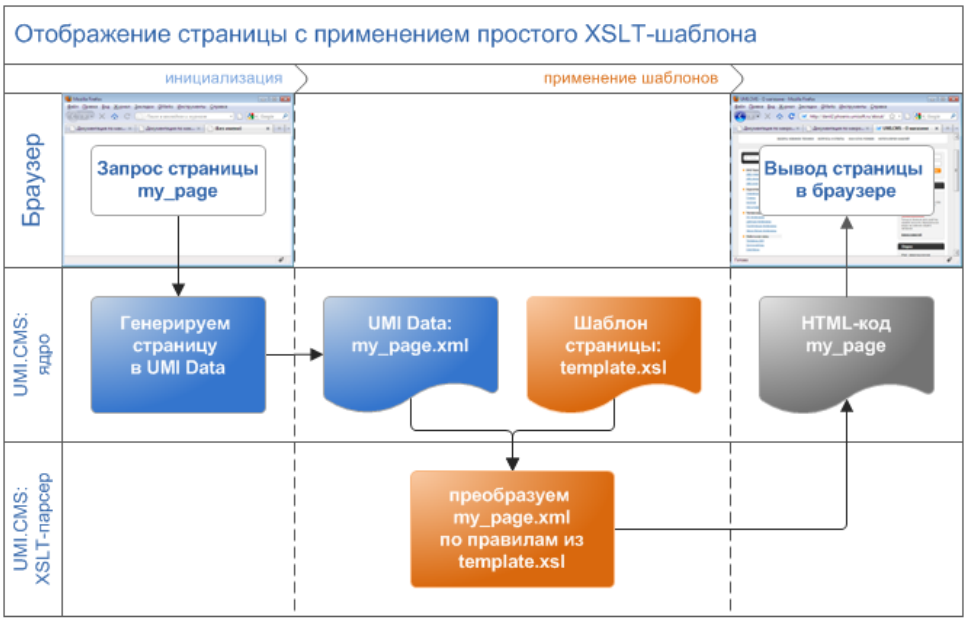


Рис. 2.6. Алгоритм отображения страницы с применением простого XSLT-шаблон

Причин для такого выбора не мало:

1) XSLT – индустриальный стандарт, поддерживаемый W3C. Во всем мире XSLT уже давно воспринимается как стандарт верстки, а в России он широко используется в компании «Яндекс»;
2) жесткое разделение бизнес-логики и модели представления данных, что позволяет придерживаться правил шаблона проектирования MVC;
3) для типовых операций достаточно создать шаблон только один раз и можно использовать его в различных частях проекта;
4) правка XSLT-шаблона не предполагает вмешательства в бизнес-логику и анализ структуры связей, которые могли бы использоваться в шаблонах других типов;
5) обмен контентом с другими ресурсами на основе XML-формата позволяет использовать сторонние сервисы на собственном сайте и при этом не зависеть от внутренней бизнес-логики этих сервисов;
6) XSLT-трансформация – один из наиболее быстрых шаблонизаторов при сравнимых объемах данных.

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

3.1. Разработка системы

После проектирования базы данных исходя из логической модели была создана даталогическая модель, которая показана на рисунке 3.1.

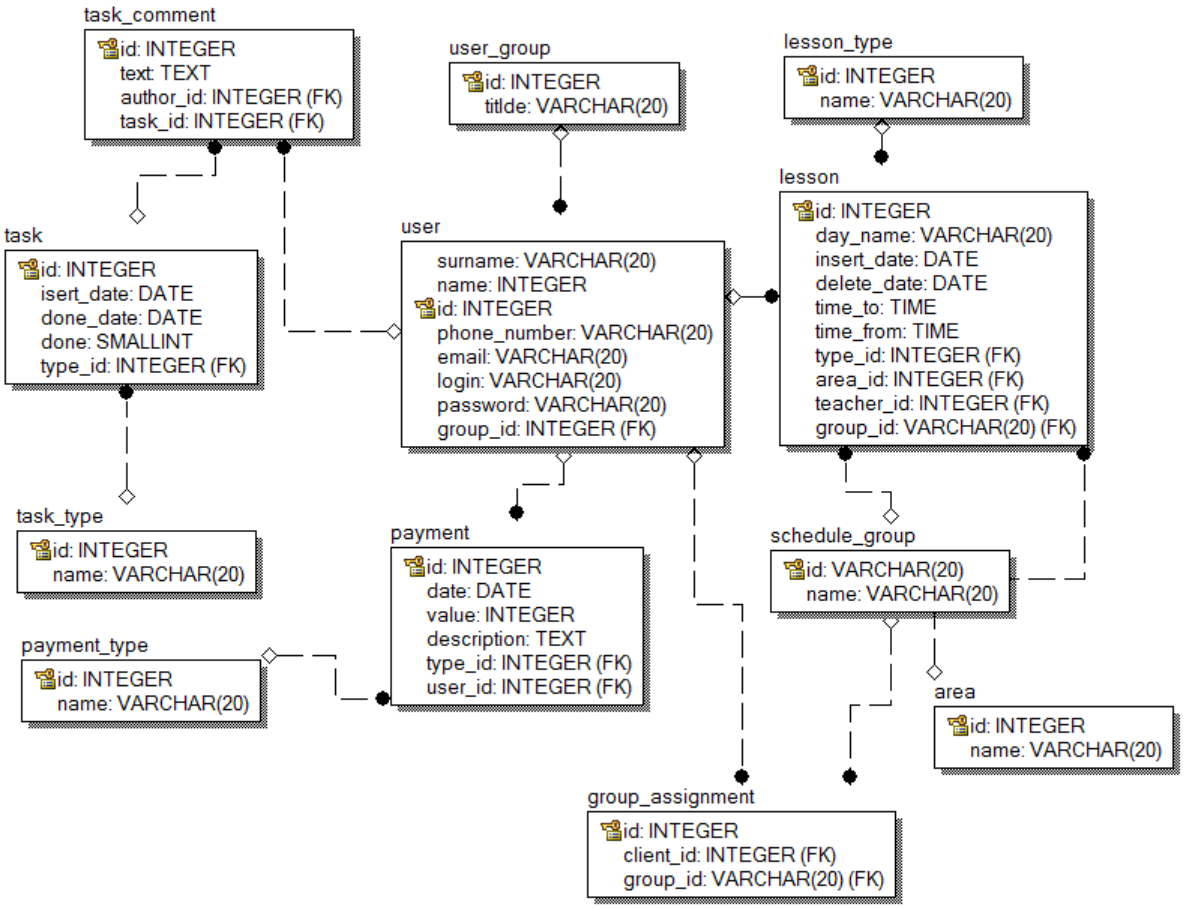


Рис. 3.1. Даталогическая схема базы данных

Для работы с базой данных было создано два класса: «Database» и «Orm». В свою очередь класс «Database», как и некоторые другие, придерживается шаблона проектирования «Singleton», основное значение которого заключается в гарантии существования только одного экземпляра класса и глобального доступа из любой части проекта [20].

Класс «Orm» по большей степени является прослойкой между объектами системы и базой данных, а также служит конструктором SQL

запросов. Методы класса делятся на 2 типа – задающие параметры запроса и исполнительные. Листинг класса Database, отвечающего за подключение к базе данных, приведен в листинге 3.1.

Листинг 3.1. Класс подключения к базе данных

```
class Database {
        private static $db = null;
        public static function getConnect(){
           if(self::$db == null) {
                global $CFG;
                $pdoString = "mysql:";
                $pdoString .= "host=".$CFG->dbhost.";";
                $pdoString .= "dbname=".$CFG->dbname;
                self::$db = new PDO($pdoString, $CFG->dbuser, $CFG->dbpass);
                self::$db->query( "SET CHARSET utf-8");
            }
            return self::$db;
        }
}
```

Метод findAll продемонстрирован в листинге 3.2 и отвечает за поиск записей в базе данных по заданным критериям и преобразовывает их в экземпляры классов.

Листинг 3.2. Метод findAll класса Orm

```
public function findAll() {
        $this->setQueryString();
        $result = Database::getConnect()->query($this->queryString);
        $result->setFetchMode(PDO::FETCH_CLASS, get_class($this));
        return $result->fetchAll();
}
```

Следующий метод getCount возвращает количество записей в базе данных (см. листинг 3.3).

Листинг 3.3. Метод getCount класса Orm

```
public function getCount() {
    $this->select = "count(".$this->getTableName().".id) as count";
    $this->setQueryString();
    $result = Database::getConnect()->query($this->queryString);
    if(!$result)    return 0;
    $result = $result->fetch();
    return intval($result['count']);
}
```

Метод save одновременно отвечает за добавление и обновление записей в базу данных. Код данного метода расположен в приложении 1.

За удаление объектов (записей) из базы данных служит метод delete. Исходный код данного метода продемонстрирован в листинге 3.4.

Листинг 3.4. Метод delete Класса Orm

```
public function delete($obj = null) {
    if(is_null($obj))   $obj = $this;
    $sTableName = $obj->getTableName();
    $query = "DELETE FROM " . $sTableName . " WHERE id = " . $obj->getId();
    $this->executeQuery($query);
}
```

3.2. Особенности процесса разработки

В процессе разработки из-за особенностей проектирования и возможностей языков пришлось столкнуться с некоторыми сложностями в реализации различных частей системы.

В языке программирования PHP, к сожалению, отсутствует стандартная реализация обработчиков событий, как в Javascript. Данный функционал позволяет расширять возможности системы без вмешательства в обработчики других классов.

Каждый обработчик содержит название события и функцию, выполняемую при срабатывании события. Для решения данной задачи было создано 3 метода для работы с наблюдателями. Работа с наблюдателями имеет довольно простой алгоритм и является универсальным решением для проектов, написанных на языке PHP [20].

Формат добавления, удаления и хранения наблюдателей напоминает стек. Обработчики одного и того же события срабатывают по порядку их добавления.

Первый метод attachObserver отвечает за добавление обработчика для определенного события. Исходный код данного метода продемонстрирован в листинге 3.5. Для добавления наблюдателя необходимо указать название события, на которое будет производиться реакция и безымянную функцию, вызываемую при возникновении события.

Листинг 3.5. Метод attachObserver

```
static public function attach($event, $handler) {
   self::$observers[$event][] = $handler;
}
```

Второй метод notify (см. листинг) устанавливается в месте срабатывания события и принимает в качестве аргументов массив – список

параметров, передаваемых в безымянную функцию метода attachObserver и название события.

Листинг 3.6. Метод notify

```
static public function notify($args, $action) {
   foreach (Core::$observers as $name => $observers)
      if($name == $action)
         foreach ($observers as $name => $function) {
            $function($args);
         }
}
```

Третий метод detachObserver просто удаляет «верхний» наблюдатель на определенной действие из общего стека наблюдателей.

К примеру, для установки наблюдателей на удаление какого-либо объекта, для начала необходимо переопределить метод delete у объекта и установить наблюдатели. Пример вызова метода notify показан в листинге 3.8. Далее, как показано в листинге 3.7, необходимо добавить обработчик на определенное событие.

Листинг 3.7. Пример обработчика для события

```
Core::attachObserver("beforeUserDelete", function($args){
   $oUser = $args[0];
   if($oUser->groupId() == 4) {
      $listValue = Core::factory("Property_List_Values")
         ->where("property_id", "=", 21)
         ->where("value", "like", "%".$oUser->name()."%")
         ->where("value", "like", "%".$oUser->surname()."%")
         ->find();

      if($listValue) $listValue->delete();
   }
   Core::factory("Property")->clearForObject($oUser);
});
```

Листинг 3.8. Пример использования метода notify

```
public function delete($obj = null) {
   Core::notify(array(&$this), "beforeUserDelete");
   parent::delete();
   Core::notify(array(&$this), "afterUserDelete");
}
```

Как видно из примеров выше, данное решение довольно простое в использовании, является универсальным и легко интегрируется в любую другую систему.

Еще одной особенностью была просьба клиента реализовать возможность сортировки записей в таблицах по столбцам, чтобы существенно облегчить работу с большим количеством записей для менеджеров. Особенно удобно так выделять должников в списке клиентов.

Изначально планировалось реализовать данный функционал при помощи AJAX. То есть отправлять запрос на сервер, который, в свою очередь, отправит запрос к базе данных на получение записей в определенном порядке, после чего перезагрузить таблицу целиком. Однако это бы создавало большую нагрузку на сервер, базу данных и выполнялось бы с довольно существенной задержкой для пользователя, особенно при работе с большими таблицами, поэтому данный вариант является неоптимальным.

Для данной задачи было найдено готовое решение «TableSorter», которое представляет из себя полноценную библиотеку, реализованную на языке Javascript. Большим плюсом данной библиотеки является то что она не задействует ресурсы сервера, все вычисления проводит на стороне клиента что позволяет производить сортировку практически моментально.

Ещё одним большим плюсом данного решения является простая интеграция в проект и универсальность. Все, что необходимо для внедрения

решения в проект это подключение основных js файлов и буквально всего одна команда по загрузке страницы как показано в листинге 3.9.

Листинг 3.9. Функция для добавления возможности сортировки таблицы

```
$(function(){
   $("#sortingTable").tablesorter();
});
```

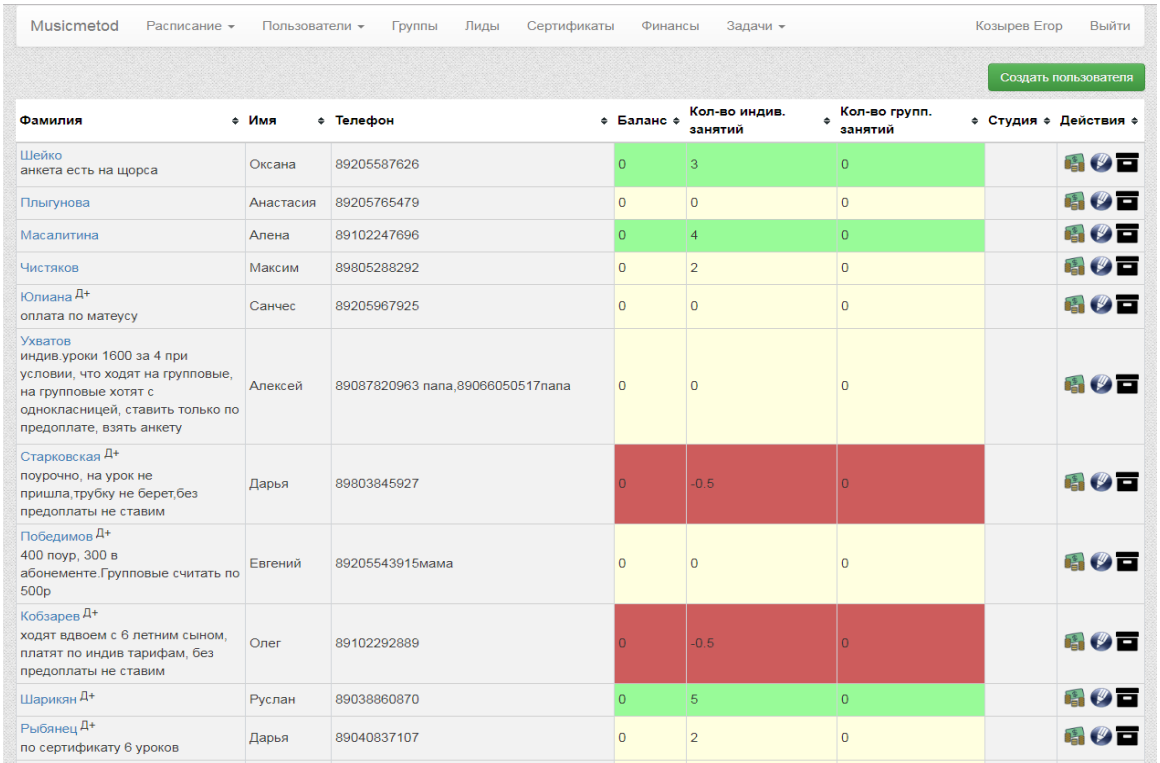
После загрузки страницы нажимая на заголовки столбцов таблицы будет происходить сортировка данных «по возрастанию», а при повторном нажатии – «по убыванию».

3.3. Тестирование

Основной частью музыкальной школы являются ученики (клиенты). Раздел клиентов устроен таблицей, содержащей краткую информацию об ученике с возможностью сортировки по столбцам к примеру, можно отсортировать клиентов по столбцу количества занятий по возрастанию, чтобы легко и быстро переместить всех должников на верх таблицы. Это существенно упрощает работу менеджеров с клиентами. Данная таблица продемонстрирована на рисунке 3.2.

Из данного раздела также доступна возможность переноса клиентов в архив. Учетные записи, находящиеся в архиве, сохраняют все данные о клиенте, однако не имеется возможности ставить таких клиентов в расписание, начислять им платежи и так далее.

Добавление нового клиента или редактирование происходит без перезагрузки страницы при помощи «всплывающего окна». Если происходит редактирование существующей записи, то в соответствующих полях уже будет внесена информация, как показано на рисунке 3.4.



Фамилия	Имя	Телефон	Баланс	Кол-во индив. занятий	Кол-во групп. занятий	Студия	Действия
Шейко анкета есть на щорса	Оксана	89205587626	0	3	0		
Плыгунова	Анастасия	89205765479	0	0	0		
Масалитина	Алена	89102247696	0	4	0		
Чистяков	Максим	89805288292	0	2	0		
Юлиана Д+ оплата по матеусу	Санчес	89205967925	0	0	0		
Ухватов индив.уроки 1600 за 4 при условии, что ходят на групповые, на групповые хотят с одноклассницей, ставить только по предоплате, взять анкету	Алексей	89087820963 папа,89066050517папа	0	0	0		
Старковская Д+ поурочно, на урок не пришла,трубку не берет,без предоплаты не ставим	Дарья	89803845927	0	-0.5	0		
Победимов Д+ 400 поур, 300 в абонементе.Групповые считать по 500р	Евгений	89205543915мама	0	0	0		
Кобзарев Д+ ходят вдвоем с 6 летним сыном, платят по индив тарифам, без предоплаты не ставим	Олег	89102292889	0	-0.5	0		
Шарикян Д+	Руслан	89038860870	0	5	0		
Рыбянец Д+ по сертификату 6 уроков	Дарья	89040837107	0	2	0		

Рис. 3.2. Таблица клиентов

Также из данного раздела доступна возможность начисления любой суммы на баланс клиентов, происходит это аналогично редактированию данных пользователя во всплывающем окне, где указывается сумма начисления, его тип и примечание, как показано на рисунке 3.3.

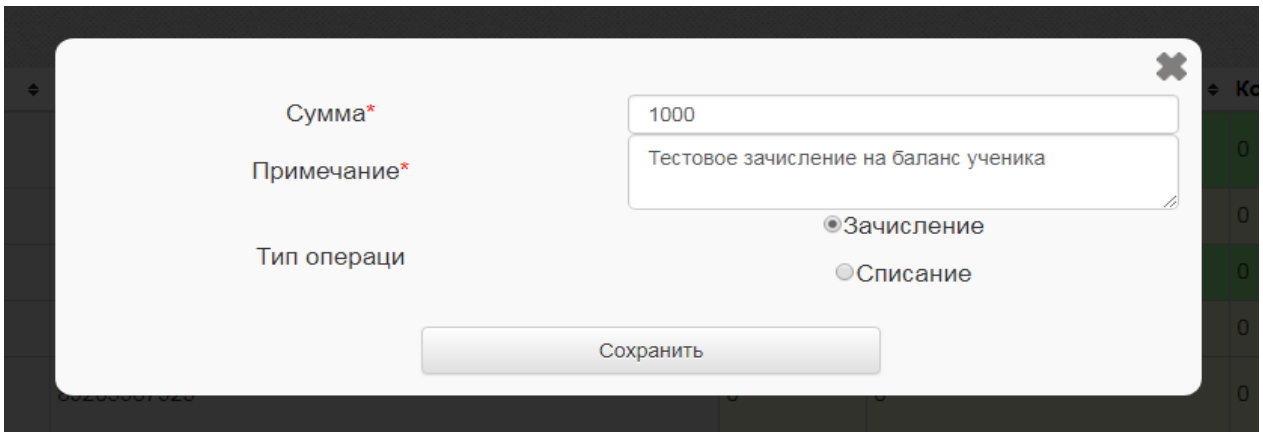


Рис. 3.3. Форма начисления баланса на счет клиента

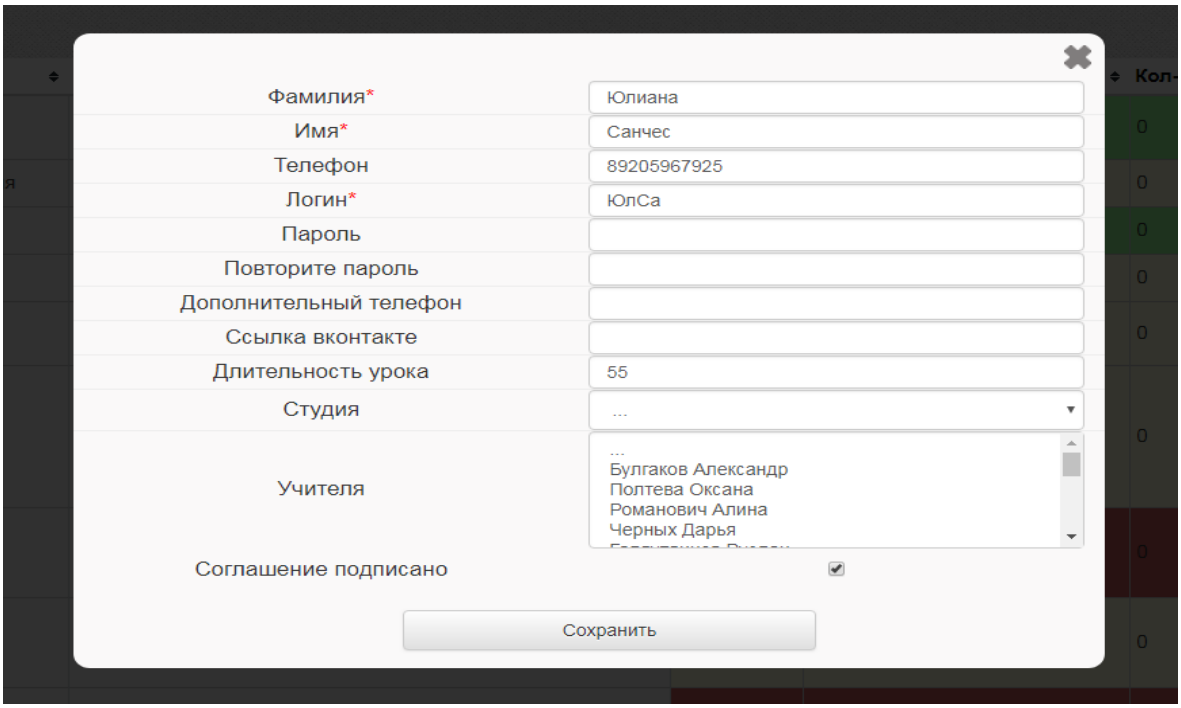


Рис. 3.4. Форма редактирования данных клиента

Результат данной операции можно посмотреть в личном кабинете ученика, который показан на рисунке 3.4.

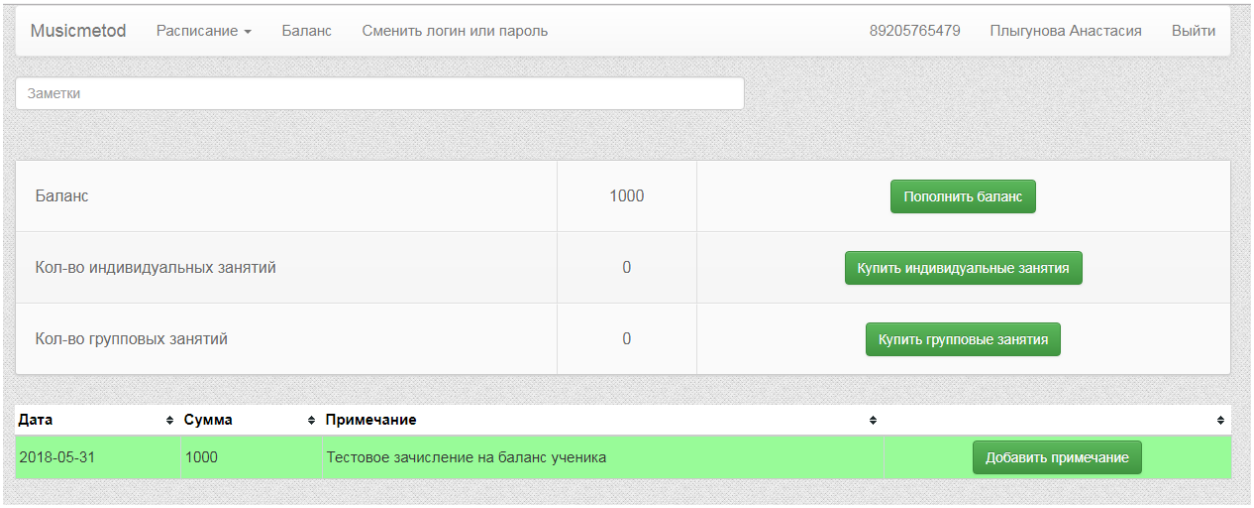


Рис. 3.4. Раздел «Баланс» в личном кабинете пользователя

Раздел «Финансы» представлен на рисунке 3.5. По большей степени данный раздел представляет из себя таблицу, содержащую информацию о платежах и инструмент для указания периода, в рамках которого выполнялись финансовые манипуляции. На рисунке 3.5 выделен тот

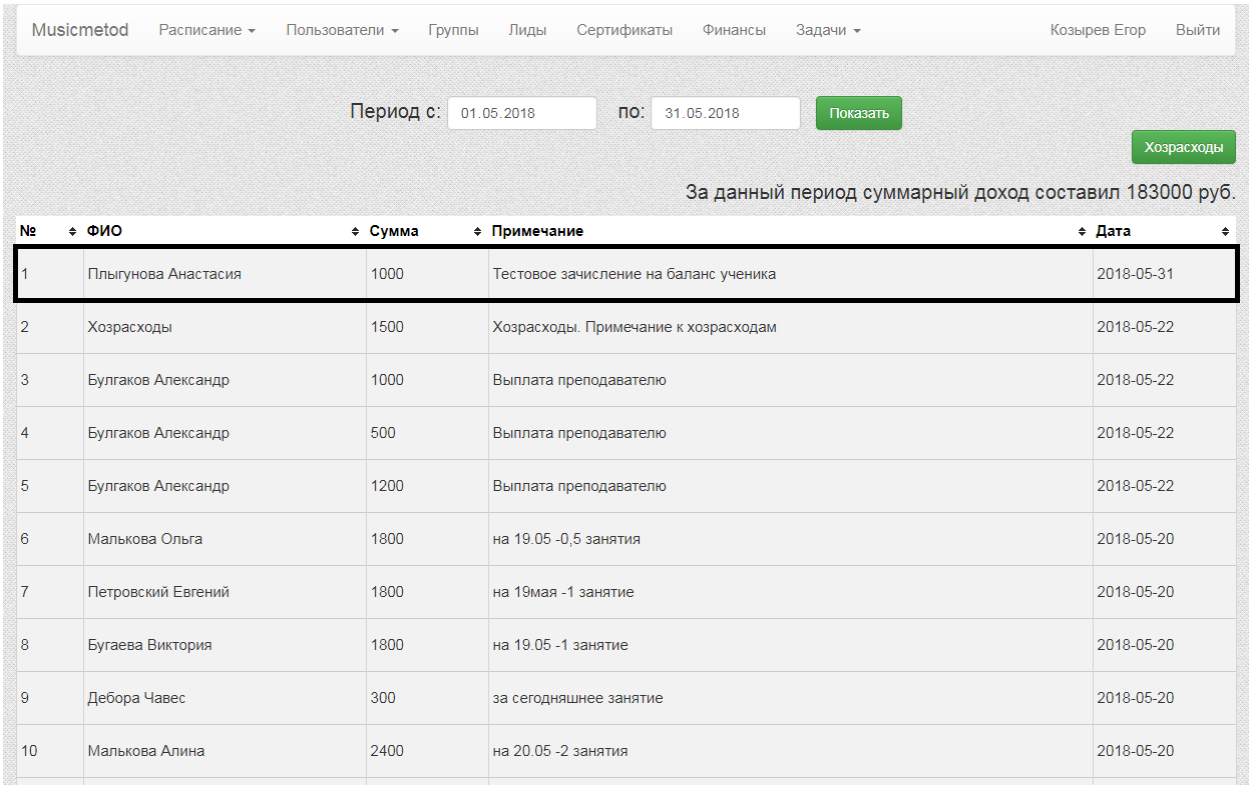
тестовый платеж, который был продемонстрирован ранее на рисунках 3.3 и 3.4. По умолчанию временной период равен одному месяцу по текущее число, однако можно задавать любые временные промежутки.

На данный момент в системе существует 4 типа выплат:

– зачисление;
– списание;
– выплата преподавателю;
– хозрасходы.

В данной таблице присутствуют платежи всех типов, кроме списаний, так как по мнению директора школы подобные записи в данной таблице не будут нести никакой полезной информации.

Учительские выплаты доступны в личных кабинетах персонала, но лишь менеджер и администратор имеют возможность добавлять выплаты.



№	ФИО	Сумма	Примечание	Дата
1	Плыгунова Анастасия	1000	Тестовое зачисление на баланс ученика	2018-05-31
2	Хозрасходы	1500	Хозрасходы. Примечание к хозрасходам	2018-05-22
3	Булгаков Александр	1000	Выплата преподавателю	2018-05-22
4	Булгаков Александр	500	Выплата преподавателю	2018-05-22
5	Булгаков Александр	1200	Выплата преподавателю	2018-05-22
6	Малькова Ольга	1800	на 19.05 -0,5 занятия	2018-05-20
7	Петровский Евгений	1800	на 19мая -1 занятие	2018-05-20
8	Бугаева Виктория	1800	на 19.05 -1 занятие	2018-05-20
9	Дебора Чавес	300	за сегодняшнее занятие	2018-05-20
10	Малькова Алина	2400	на 20.05 -2 занятия	2018-05-20

Рис. 3.5. Раздел финансы

Ещё одним важным моментом было то, что было бы очень удобно, по мнению директора, выводить на странице общую сумму дохода за данный период, что в свою очередь тоже можно увидеть на рисунке 3.5.

Раздел «задачи» делиться на 3 подраздела:

– общий список задач;
– список задач «на сегодня»
– актуальные задачи.

В первом подразделе довольно большая таблица, состоящая из всех когда-либо существовавших задач.

Следующий подраздел состоит также из таблицы задач, которые имеют тип «на сегодня» и «дату контроля» начиная с текущей.

Самым важным и используемым является третий подраздел «актуальные задачи». В этой таблице собраны задачи, имеющие тип «на сегодня» и текущую дату контроля, а также все остальные невыполненные задачи, имеющие тип «до выполнения». Таблица из данного подраздела продемонстрирована на рисунке 3.6. К задачам менеджеры могут добавлять комментарии. Выполненные задачи исчезнут из списка на следующий день.

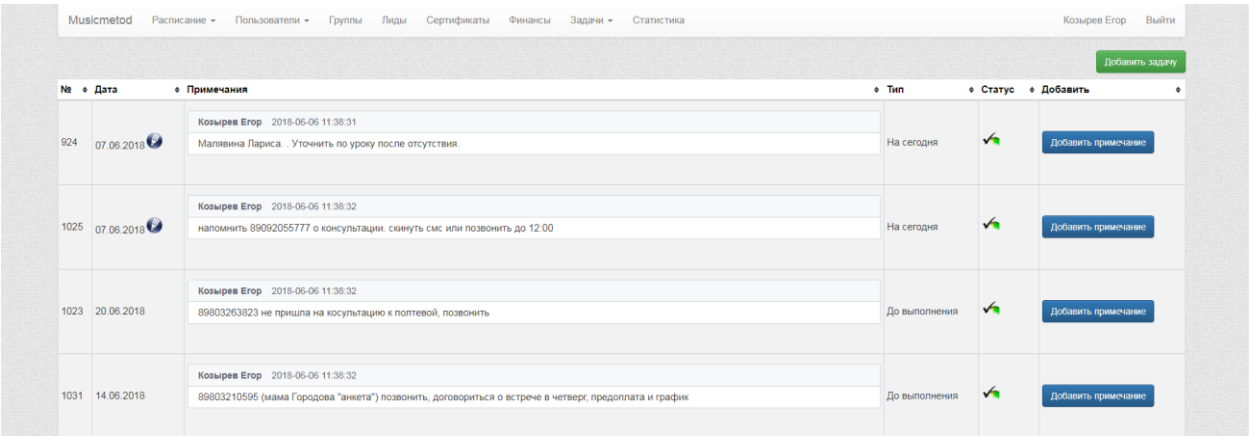


Рис. 3.6. Подраздел «Актуальные задачи» раздела «Задачи»

Создание задачи происходит в модальном окне без перезагрузки страницы, как показано на рисунке 3.7.

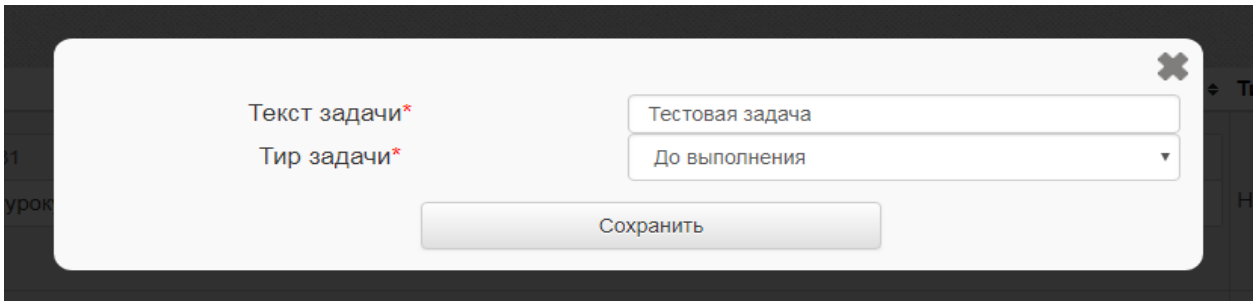


Рис. 3.7. Пример создания задачи

После нажатия на кнопку «Сохранить» модальное окно закрывается и таблица задач перезагружается при помощи AJAX. Результат добавления задачи показан на рисунке 3.8.

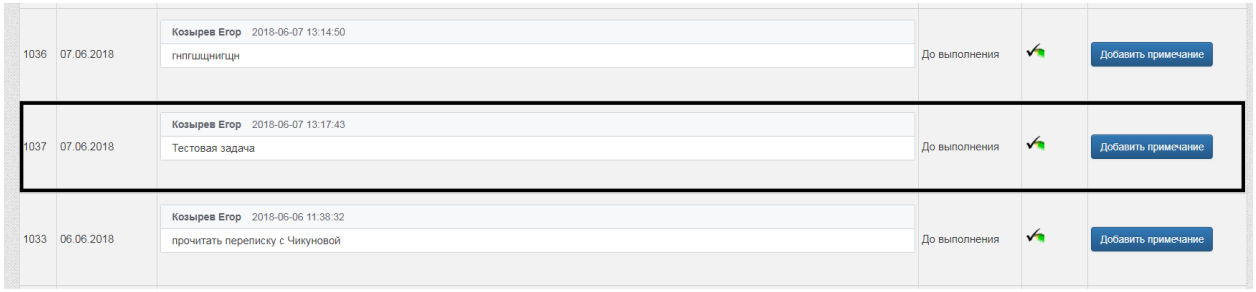


Рис. 3.8. Результат добавления задачи

К тому же у задач может быть неограниченное количество комментариев, которые содержат информацию об авторе комментария и даты со временем создания. Создается комментарий в модальном окне, сохраняется без перезагрузки страницы и сражу обновляется содержание таблицы. Пример задачи с несколькими комментариями показан на рисунке 3.9.

Рис. 3.9. Пример задачи с несколькими комментариями

Следующий раздел «Статистика» доступен только директору (администратору). Он продемонстрирован на рисунке 3.10 и содержит в себе краткую сводку о количестве проведенных занятий, посещаемость, общая сумма выплат преподавателям и информацию по количеству и статусам лидов за указанный период. Как и в финансах, по умолчанию период составляет один месяц включая текущую дату.

Первая таблица содержит в себе информацию об общей сумме баланса всех учащихся, а также количество оплаченных и неоплаченных занятий в независимости от указанной даты.

Раздел сертификаты очень прост и ни с чем другим не связан и содержит кнопку для добавления новой записи и таблицу с данными о выданных сертификатах. Добавление происходит аналогично добавлению данных в других разделах при помощи модального окна и без перезагрузки страницы.

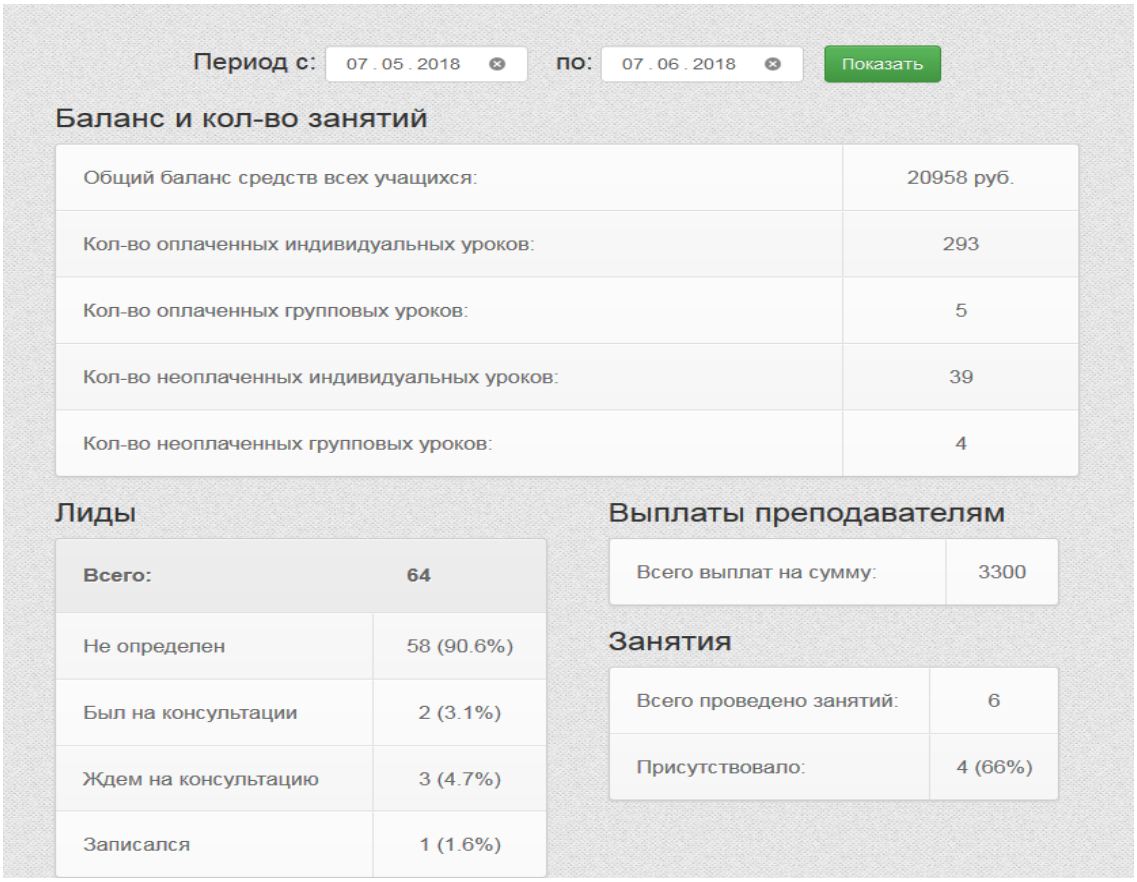


Рис. 3.10. Раздел «Статистика»

Самый используемый раздел это «Расписание». Клиентам и учителям в расписании доступны лишь те занятия, в которых они задействованы, а менеджеры и директор имеют доступ к «полной картине». Занятия подсвечены цветами, которые соответствуют ученикам в списке клиентов. Для просмотра расписания в определенный день над таблицей расположен календарь. При смене даты перезагружается таблица с расписанием на указанную дату. На рисунке 3.11 изображен фрагмент расписания, на котором обозначены 4 особенности данной системы:

– 1 пример занятия с клиентом, у которого выставлен период отсутствия;
– 2 пример разового отсутствия клиента на занятии;
– 3 пример разового занятия;
– 4 пример изменения времени проведения занятия.

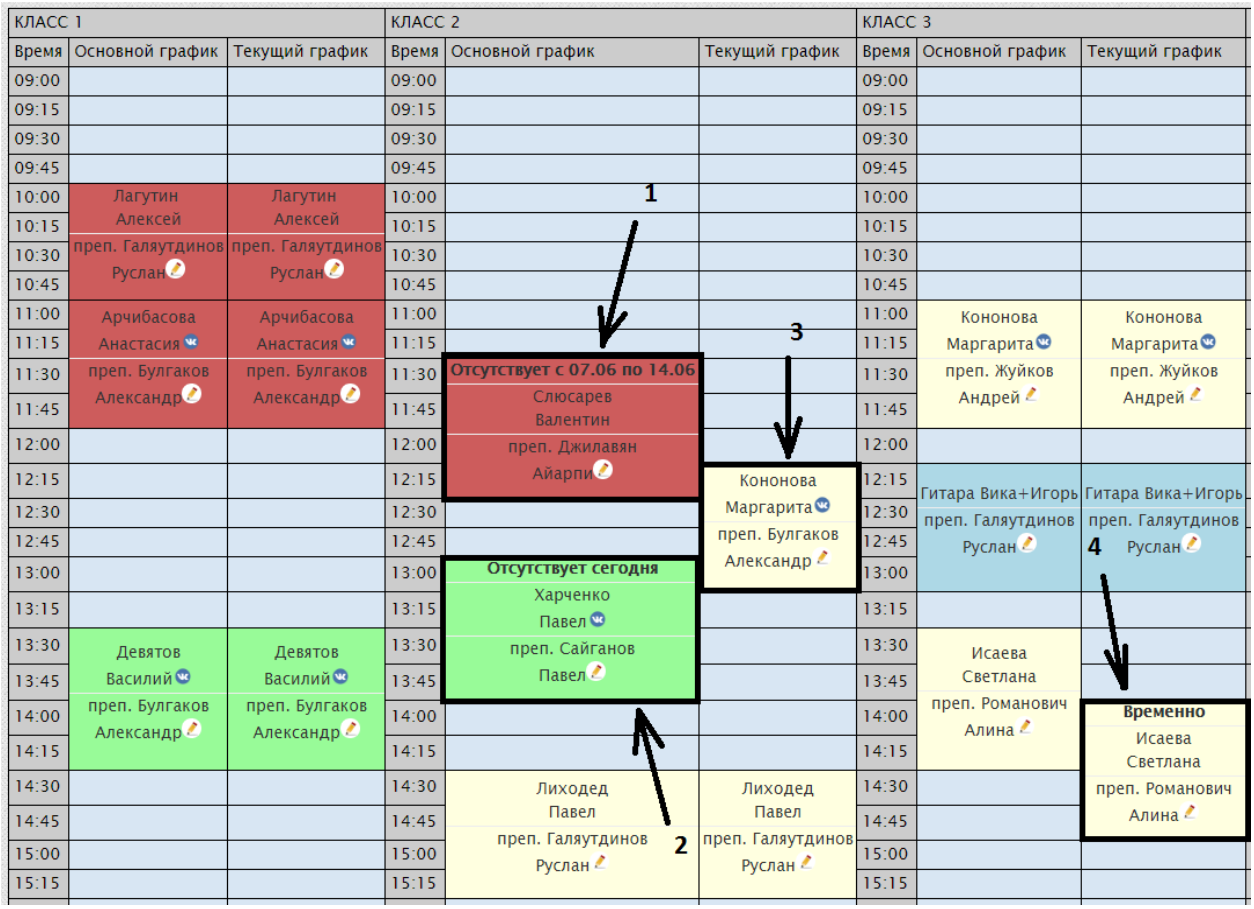


Рис. 3.11. Фрагмент таблицы расписания

Самым важным и функциональным является административный раздел самого ядра системы, из которого есть доступ абсолютно ко всем данным системы и данным музыкальной школы. Переключение между вкладками раздела, а также переход по любым ссылкам происходит без перезагрузки страницы посредством технологии AJAX. По такому же принципу происходит удаление, добавление, редактирование данных.

К системным данным относятся:

– пользователи;
– структуры;
– макеты;
– константы;
– дополнительные свойства;
– списки;
– формы редактирования;
– пункты административного меню.

На рисунке 3.12 показана вкладка «Пользователи». Также на рисунке 3.12 отмечены 3 кнопки:

– 1 работа с дополнительными свойствами группы;
– 2 редактирование информации о группе;
– 3 удаление группы пользователей.

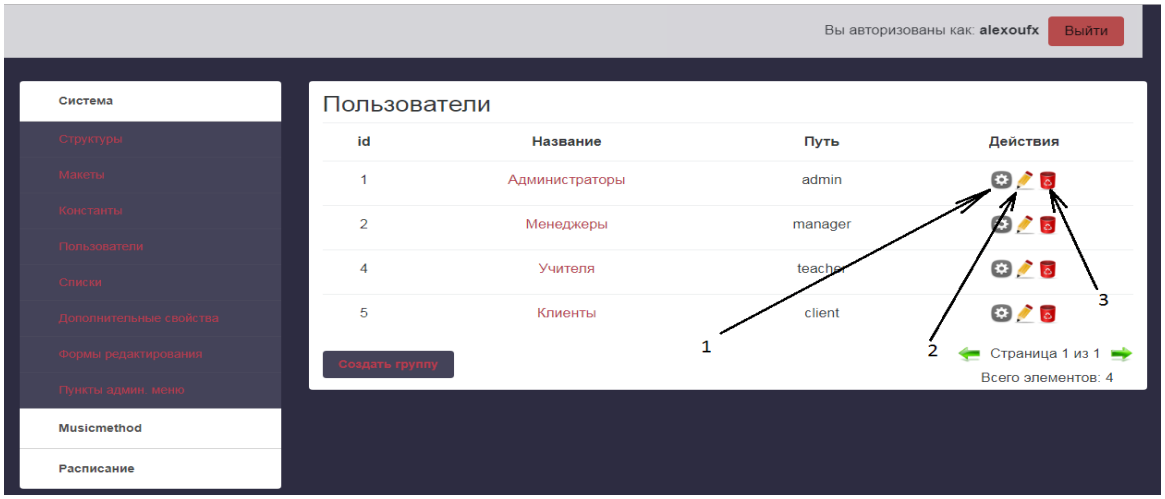


Рис. 3.12. Вкладка «Пользователи» административного раздела ядра системы

Аналогично выглядят таблицы для других объектов. Форма редактирования объекта формируется динамически. Пример редактирования объекта приведен на рисунке 3.13. Всем основным полям формы редактирования соответствуют записи во вкладке «Формы редактирования» которая показана на рисунке 3.14.

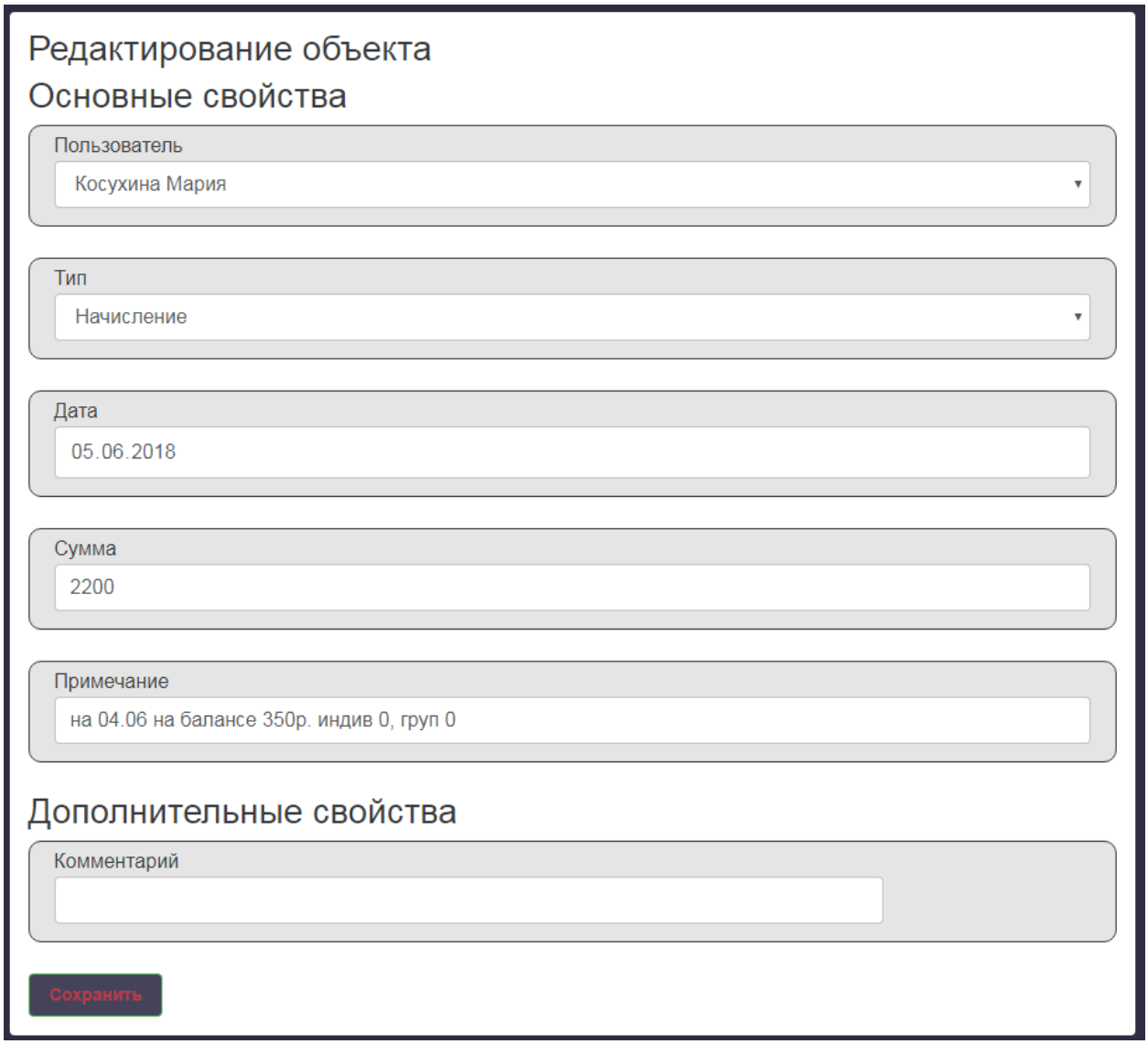


Рис. 3.13. Форма редактирования платежа

Как можно заметить на рисунке 4.14 поля формы редактирования объекта можно редактировать, удалять, добавлять и делать их активными или неактивными. Поля для редактирования дополнительных свойств формируются на основании названий и типов данных свойств.

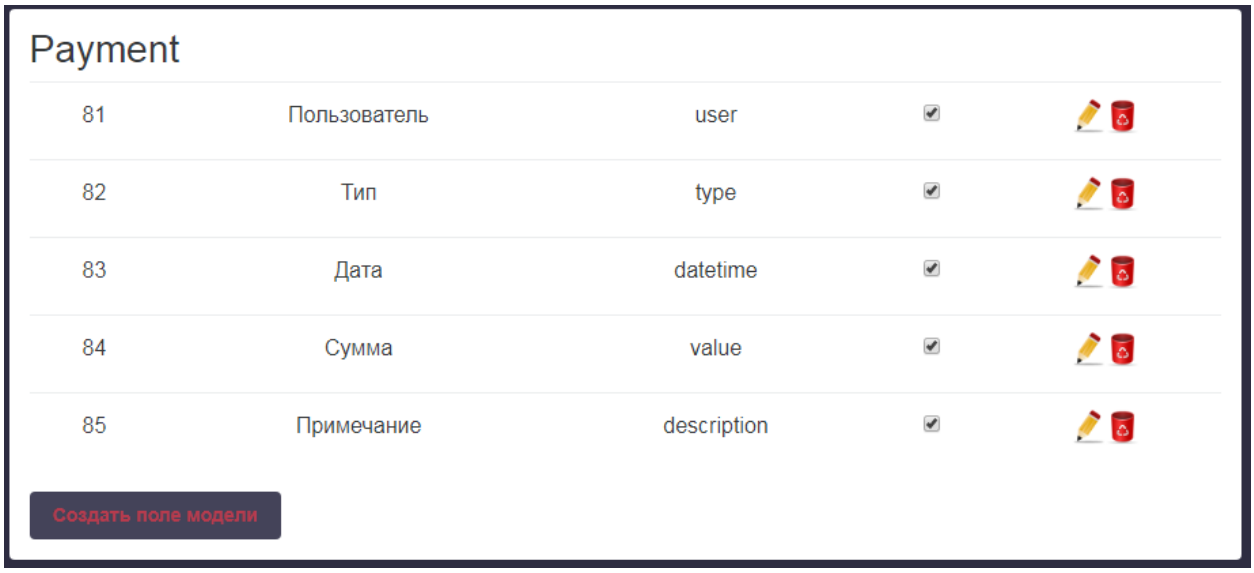


Рис. 3.14. Список полей для формы редактирования платежа

Дополнительные свойства могут принадлежать абсолютно любому объекту. Этот функционал системы позволяет создавать различные свойства объекта при помощи разработанного интерфейса административного раздела без вмешательства в таблицы базы данных и классы. Дополнительные свойства могут быть активны или неактивны, иметь значение по умолчанию, а также тип значения свойства:

– число (float);
– флажок (bool);
– строка (varchar);
– текст (text);
– список;

В следующих версиях системы будут добавлены типы:

– файл (varchar);
– дата (date);
– время (time);

Значение дополнительного свойства типа «Список» содержит указатель на элемент списка. Списки создаются и редактируются во вкладке: «Списки».

Сначала необходимо создать дополнительное свойство во вкладке «Дополнительные свойства», а затем назначить его для группы или любого родительского элемента, в данном случае это группа пользователей «Клиенты». Пример работы с доп. свойствами группы показан на рисунке 3.15, на котором можно увидеть таблицу с созданными свойствами, где указаны их: название, описание, тип и принадлежность разделу.

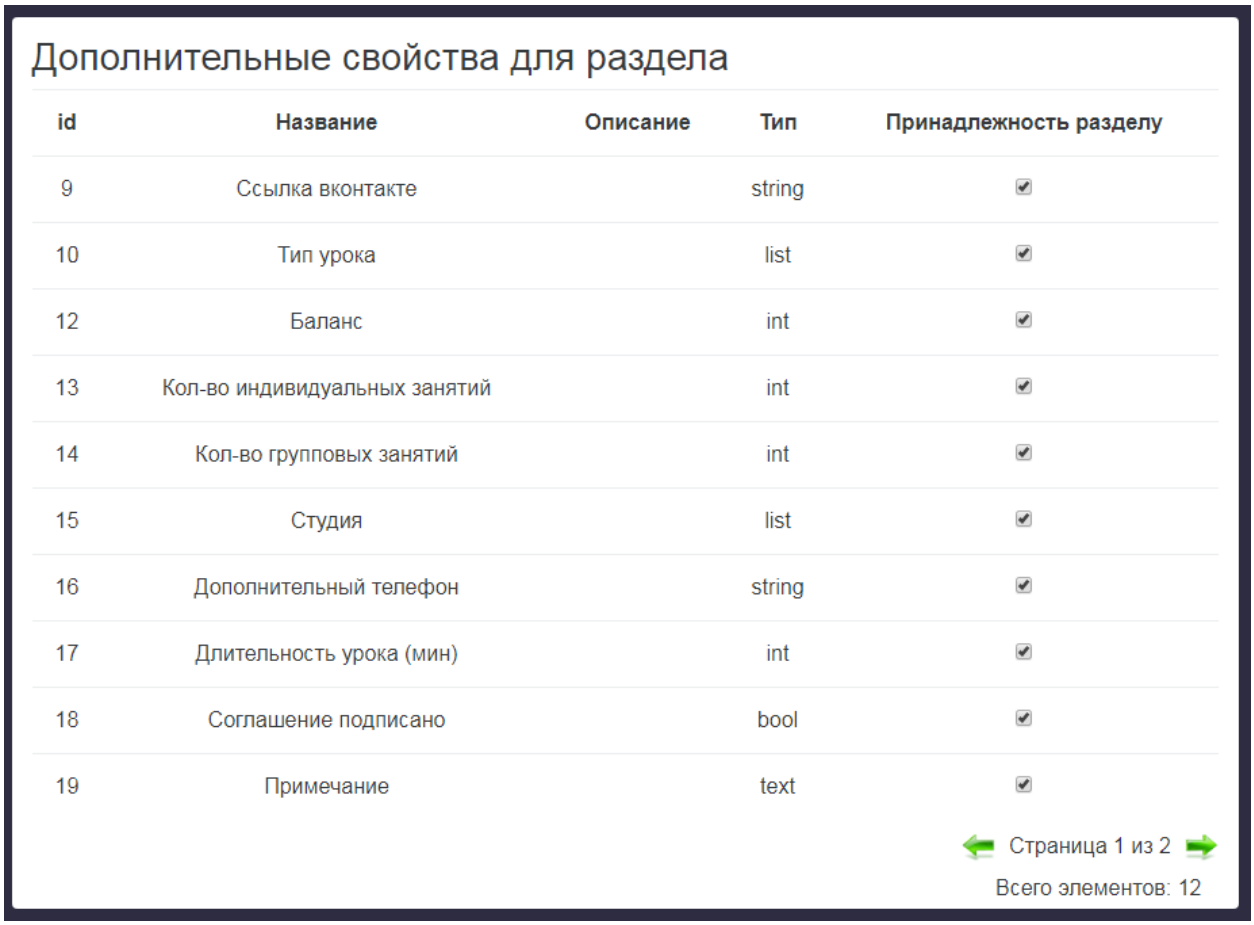


Рис. 3.15. Интерфейс для работы с доп. свойствами группы пользователей

ЗАКЛЮЧЕНИЕ

В рамках ВКР была создана система учета для частной музыкальной школы «Musicmetod», позволяющая централизовать и существенно упростить рабочий процесс для менеджеров и директора за счет простого и удобного функционала. В ходе работы были выполнены все поставленные задачи, а именно:

- возможность добавления/редактирования занятий в расписании;
- управление данными клиентов и преподавателей;
- создание, редактирование и покупка тарифов;
- работа с лидами;
- вывод краткой отчетности (статистика);
- управление балансом клиентов и контроль над любыми финансовыми операциями в пределах учреждения;
- способ коммуникации между управляющим персоналом посредством добавления задач/комментариев к задачам;
- возможность клиентов распоряжаться собственным балансом для покупки тарифов;

На данный момент система полностью функционирует и удовлетворяет всем требованиям заказчика. Уже ведется обсуждение доработок и расширения функционала для большей автоматизации рабочего процесса.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. AJAX и PHP. Разработка динамических веб-приложений / Богдан Бринзаре, Кристиан Дари, Михай Бусика, Филип Черчез-Тоза. – Москва: Символ-Плюс, 2009. – 336 с.
2. Бейтс Б. Паттерны проектирования / Б. Бейтс, К. Сиера, Э. Фримен, Э. Фримен. – Санкт Петербург: Питер 2013. – 655 с.
3. Бенедетти Р. Изучаем работу с jQuery. – Издательский дом" Питер", 2012.
4. Гамма Э. и др. Приемы объектно-ориентированного проектирования. – " Издательский дом"" Питер""", 2013.
5. Голицина О.Л. Основы проектирования баз данных. Учебное пособие / О.Л. Голицина, Т.Л. Партыкова, И.И. Попов. – Москва: Форум 2014. –416 с.
6. Грегер С. Э., Сковородин Е. Ю. Построение онтологического портала с использованием объектной базы данных //Объектные системы. – 2010. – №. 1 (1).
7. Дакетт Д. Javascript и Jquery. Интерактивная веб-разработка / Д. Дакетт – Москва: Эксмо, 2017. – 640 с.
8. Дронов В. А. HTML 5 и CSS 3: Современный Web-дизайн. – БХВ-Петербург, 2011.
9. Дронов В. А. JavaScript и AJAX в Web-дизайне. – БХВ-Петербург, 2012.
10. Зандстра М. PHP. Объекты, шаблоны и методики программирования / М. Зандстра. – Москва: Вильямс 2016. – 576 с.
11. Илюшечкин В. Основы использования и проектирования баз данных. – Litres, 2018.
12. Котеров Д.В. PHP 7 / Д.В. Котеров, И.В. Смидянов. – Санкт Петербург: БХВ-Петербург 2017. –265 с.
13. Кузнецов М. В. Объектно-ориентированное программирование на PHP. – БХВ-Петербург, 2012.

14. Кэй М. XSLT. Справочник программиста / М. Кэй. – Москва: Символ-Плюс 2002. –1013 с.
15. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство первое издание / Маклафлин Б. – Санкт Петербург 2004.
16. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд //СПб.: Питер. – 2013. – 2013.
17. Симонова О. Н., Лясин Д. Н. Шаблон проектирования MVC как эффективное средство построения архитектуры программной системы //Современные наукоемкие технологии. – 2014. – №. 5-2. – С. 96-97.
18. Тахагхогхи С. Руководство по MySQL / С. Тахагхогхи, Хью Е. Вильямс. – Москва: Русская Редакция, 2007. – 544 с.
19. Тидуэлл Д. XSLT / Д. Тидуэлл. – Москва: Символ-Плюс 2013. –960 с.
20. Фримен Э. и др. Паттерны проектирования //СПб.: Питер. – 2011.
21. Харрис Э. PHP/MySQL для начинающих. – М. : ИД Кудиц-Образ, 2005.
22. Хрусталев А.А. HTML5 + CSS3. Основы современного WEB-дизайна / А.А. Хрусталев, А.В. Кириченко. – Москва 2018. –352 с.
23. Chaudhary M., Kumar A. PhpStorm Cookbook. – Packt Publishing Ltd, 2014.
24. Listok CRM – автоматизированная система учета посещений, абонементов и записей на занятия [Электронный ресурс] // ООО “ЛИСТОК АЙТИ” –2018. Режим доступа: https://listokcrm.ru, свободный.
25. Hollihop – система для управления учебным центром [Электронный ресурс] // Режим доступа: https://holyhope.ru, свободный.

ПРИЛОЖЕНИЕ 1

Метод save класса Orm

```
public function save() {
	$objData = $this->getObjectProperties();
	$aRows = array_keys($objData);
	$aValues = array_values($objData);

	if($this->id) {
		$queryStr = "UPDATE ".$this->getTableName()." ";
		$queryStr .= "SET ";

		for($i = 0; $i < count($objData); $i++) {
			$i + 1 == count($objData)
				? $queryStr .= "`".$aRows[$i]."` = '".$aValues[$i]."' "
				: $queryStr .= "`".$aRows[$i]."` = '".$aValues[$i]."', ";
		}
		$queryStr .= "WHERE `id` = '".$this->getId()."'";
	}
	else  {
		$queryStr = "INSERT INTO ".$this->getTableName()."(";

		for($i = 0; $i < count($objData); $i++) {
			$i + 1 == count($objData)
				? $queryStr .= $aRows[$i]
				: $queryStr .= $aRows[$i].", ";
		}
		$queryStr .= ") VALUES(";

		for($i = 0; $i < count($objData); $i++) {
			$i + 1 == count($objData)
				? $queryStr .= "'".$aValues[$i]."'"
				: $queryStr .= "'".$aValues[$i]."', ";
		}

		$queryStr .= ") ";
	}

	$result = Core_Database::getConnect()->query($queryStr);

	if(!$this->id){
		$lastInsertId = Core_Database::getConnect()->query("SELECT
LAST_INSERT_ID() as id");
	$lastInsertId->setFetchMode(PDO::FETCH_CLASS, "stdClass");
	$lastInsertId = $lastInsertId->fetch();
	$this->id = $lastInsertId->id;
	}
	return $this;
}
```

ПРИЛОЖЕНИЕ 2

Класс Core

```
class Core {
   static private $observers = [];

   private function __construct(){}
   private function __clone(){}


   /**
    * Метод добавления
    * @param $event - название события
    * @param $handler - безымянная функция (обработчик)
    */
   static public function attach($event, $handler) {
      self::$observers[$event][] = $handler;
   }

  /**
   * Удаление последнего добавленного наблюдателя для события
   * @param $event - название события
   */
   static public function detach($event) {
      foreach (self::$observers as $name => $observers)
         if($name == $event)
            array_pop(self::$observers[$name]);
   }

  /**
   * Оповещение подписчиков о событии
   * @param $event - название события
   */
   static public function notify($args, $event) {
      foreach (self::$observers as $name => $observers)
         if($name == $event)
            foreach ($observers as $name => $function){
               $function($args);
            }
   }

  /**
   * Реализация шаблона проектирования - фабрика
   * @className – название класса
   * @$id – идентификатор объекта класса, хранящийся в базе данных
   */
   static public function factory($className, $id = null) {
         $segments = explode("_", $className);
         $filePath = ROOT . "/models";
         $obj = null;
```

```
        foreach ($segments as $segment)
                $filePath .= "/".lcfirst($segment);

        if(file_exists($filePath."/model.php") && !class_exists($className."_Model")) {
                 include_once $filePath."/model.php";
        }

        if(file_exists($filePath.".php") && !class_exists($className)) {
           include_once $filePath.".php";
        }

        if(class_exists($className))
                $obj = new $className;
        else
                return false;

        return $obj;
    }

  /**
   * Получение строки по названию с передачей в неё параметров
   * @param $sMessageName - название строки
   * @param $aMessageParams – массив параметров
   */
     public static function getMessage($sMessageName, $aMessageParams = array()) {
        ini_set('display_errors','Off');
        $aStrings = include ROOT . "/config/messages/ru/messages.php";

        if(isset($aStrings[$sMessageName])) {
           return $aStrings[$sMessageName];
        }
        else {
           return $aStrings["UNDEFIND_STRING_NAME"];
        }

        ini_set('display_errors','On');
    }
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*