# Введение

Интернет сегодня занимает важную роль в нашей повседневной жизни. Существует большое количество вещей, которые можно сделать с помощью Интернета. Можно сказать, что с развитием Интернета мы прогрессируем в каждой сфере жизни, поскольку интернет не только упрощает решение многих задач, но и экономит массу времени. Сегодня Интернет используется для самых различных целей, из основных можно выделить:

1. Связь

 На данный момент самое простое, что можно делать при помощи Интернета, - это общаться с людьми, живущими далеко от нас, с чрезвычайной легкостью. Раньше такое общение было сложной задачей, а теперь человек может легко связаться с близкими, которые находятся в другой части мира. Электронная почта, социальные сети плотно вошли в нашу жизнь и облегчили ее в значительной степени.

2. Образование

 Образование - это одна из лучших вещей, которую может предоставить нам интернет. Есть ряд книг, справочников, онлайновых справочных центров и других учебных материалов, ориентированных на изучение в интернете, которые могут сделать процесс обучения более легким. В любое время вы можете посетить нужные вам Интернет-ресурсы и получить ту информацию в которой нуждаетесь.

3. Финансы

Благодаря интернету, теперь вам не нужно стоять в очереди вашего банка, чтобы выполнить какую-либо транзакцию. Теперь можно легко выполнять финансовые транзакции через Интернет, а также можно можете легко купить или продать что-либо не выходя из дома.

В окна браузеров перешли многие приложения, которые раньше требовали наличия мощного компьютера. Сегодня же их можно легко использовать имея под рукой смартфон, или не самый мощный компьютер.

***Цель дипломной работы*** – создать веб приложение с использованием языков PHP, HTML, JS, CSS и базы данных MYSQL.

***Задачи дипломной работы:***

1. изучение теоретических основ разработки веб приложений.
2. выбор платформы и, соответственно, среды программирования, на которой будет реализовано приложение;
3. реализация приложения.

***Объектом исследования*** в данной работе являются веб-приложение

***Предмет исследования:*** процесс разработки веб-приложения

В качестве языков реализации приложения выбраны php, html, css, java script, в качестве базы данных- mysql.

Работа состоит из трех глав. В первой главе приводится описание операционных систем мобильных устройств. Во второй главе рассматриваются технологии создания мобильных приложений. Глава 3 посвящена разработке мобильного приложения «Расписание общественного транспорта города Могилева».

Дипломная работа содержит && страниц, && рисунков, 1приложение, & использованных литературных источников.

# Описание предметной области

## Концепция веб-приложения «органайзер-преподавателя»

За последние несколько лет стала очень актуальна разработка веб-приложений. Человеческая жизнь всё больше переходит на просторы интернета. Сегодня мы имеем возможность хранить и обрабатывать большие объёмы информации на удалённых компьютерах, называемых серверами, посредством обычного веб-браузера так как это удобно с точки зрения доступности и достаточно надёжно.

Преподавателям приходится собирать, хранить и обрабатывать массу информации на своей работе. К такой информации можно отнести состав учебных групп, информацию об учащихся, расписании занятий и иных мероприятий на которых нужно будет присутствовать, хранить и иметь быстрый доступ к различным материалам, необходимым для обучения, приходится подсчитывать различные данные по результатам своей работы такие как средний балл или посещаемость. Для того, чтобы хоть немного упростить эти задачи было решено разработать веб-приложение, выполняющее аналогичные задачи.

Объектом разработки является приложение, которое позволит собрать необходимую информацию, сгруппировать её таким образом, чтобы пользователь в несколько кликов мог получить интересующую информацию и результат её анализа. Это позволит пользователю быстрее ориентироваться и сэкономит много времени.

## Технические особенности веб-приложений

Существенное преимущество веб-приложений заключается в том, что их функции должны выполняться независимо от операционной системы пользователя приложения. Вместо того, чтобы писать различные версии для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создаётся один раз для произвольно выбранной платформы и на ней разворачивается. Однако различная реализация HTML, CSS, DOM и других спецификаций в браузерах может вызвать проблемы при разработке веб-приложений и последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера, например, размер шрифта, цвета, отключение поддержки сценариев, может препятствовать корректной работе приложения.

Другой, менее универсальный, подход заключается в использовании Adobe Flash, Silverlight или Java-апплетов для полной или частичной реализации пользовательского интерфейса. Поскольку большинство браузеров, как правило, с помощью плагинов, поддерживает эти технологии. Так как они предоставляют программисту больший контроль над интерфейсом, они способны обходить многие несовместимости в конфигурациях браузеров, хотя несовместимость между Java- или Flash-реализациями на стороне клиента также может приводить к различным осложнениям.

Однако такой подход не очень популярен, в силу безопасности и уже с 2015 года технологию Adobe Flash и java-апплеты не поддерживают Chrome, Safari, и другие популярные браузеры.

## Паттерн MVC

Создание и модификация веб-приложения представляет собой достаточно сложную и трудоёмкую задачу, над которой, как правило трудится не один, а целая команда специалистов, каждый из которых выполняет поставленную перед ним задачу. Их эффективное взаимодействие не было бы возможным без некоторых приёмов. К одному из таких приёмов можно отнести паттерн  **MVC.**  Открытый в конце 1970-х, он представляет собой шаблон проектирования архитектуры программного обеспечения, основной задачей которого является отделение функций работы с данными от их представления. Теоретически, грамотно спроектированное MVC-приложение позволяет фронтенд и бэкенд разработчикам в ходе работы не вмешиваться в зоны ответственности друг друга.

Название паттерна происходит от аббревиатуры трёх слов: Model (модель), View (представление) и Controller (контроллер).

Модель используется для доступа и манипулирования данными. В большинстве случаев модель — это то, что используется для доступа к хранилищу данных (например, базе данных). Модель предоставляет интерфейс для поиска данных, их создания, модификации и удаления из хранилища. В контексте паттерна MVC модель является посредником между представлением и контроллером.

Крайне важной чертой модели является то, что технически она не имеет никаких знаний ни о том, что происходит с данными в контроллере и представлении. Модель никогда не должна делать или ожидать каких-либо запросов в других компонентов паттерна.

Представление

Представление - это то, где данные, полученные от модели, выводятся в нужном виде. В традиционных веб-приложениях, разработанных в рамках MVC-паттерна, представление - это часть системы, где выполняется генерация HTML-кода. Представление также отвечает за получение действий от пользователя с тем чтобы отправить их контроллеру. Например, представление отображает кнопку в пользовательском интерфейсе, а после её нажатия вызывает соответствующее действие контроллера.

Контроллер

Контроллер — это последняя часть связки MVC. Задачей контроллера является получение данных от пользователя и манипуляция моделью. Именно контроллер, и только он, является той частью системы, которая взаимодействует с пользователем.

В двух словах контроллер можно описать как сборщик информации, передающий её модели для обработки и хранения. Он не должен делать ничего с данными, а только лишь уметь получать их от пользователя.

Разработанное приложение в некоторой степени реализует паттерн MVC.

## Заключение

# Выбор и описание базовых технологий для создания веб-приложений

## Архитектура клиент-сервер

Как правило компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами, другие имеют возможность обращаться к этим ресурсам. Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса. Клиент и сервер какого-либо ресурса могут находится как на одном компьютере, так и на различных компьютерах, связанных сетью.

Сервер может обслуживать нескольких клиентов одновременно. В этом случае говорят о многопользовательском режиме.

Цикл выполнения запроса состоит из пересылки запроса и ответа между клиентом и сервером и непосредственным выполнением этого запроса на сервере.

Существует концепции построения системы клиент-сервер:

1) Слабый клиент - мощный сервер - вся обработка информации осуществляется целиком сервером. Сервер посылает готовый результат, не требующий дополнительной обработки. Клиент только ведет диалог с пользователем: составляет запрос, отсылает запрос, принимает запрос и выводит информацию.

2) Сильный клиент - часть обработки информации перепоручается клиенту.

При частичной обработке данных на клиенте "время ожидания" меньше. Меньше оно за счет упрощения запроса и времени его выполнения. Отсюда меньше ожидание в очереди для исполнения запроса.

Безусловно, что время конечной обработки на клиенте может быть немного выше за счет разности в производительности сервера. Отсюда может несколько возрастать время ожидания. В конечном счете, это все равно выгоднее, т.к. время ожидания в очереди запросов на сервере меньше.

Многие серверы не выдерживают нагрузки и просто "подвисают". В таком случае есть два альтернативных способа: увеличение производительности и перенос части операций над данными на клиента. Как правило, увеличение производительности значительно более дорогостоящая операция, и тоже, в своем смысле, конечная. Остается только "разгрузка" сервера и перенос части обработки данных на клиента.

## Веб-сервер

Большинство веб-приложений построено на клиент-серверной архитектуре с использованием веб-сервера.

Веб-сервер — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно железо, на котором это программное обеспечение работает.

Клиент, которым обычно является веб-браузер, передаёт веб-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы — это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.

На сегодняшний день среди веб-серверов самым популярным является Apache — свободный веб-сервер, наиболее часто используемый в UNIX-подобных операционных системах, однако являющийся кроссплатформенным ПО, поддерживающим операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Существует множество модулей, добавляющих к Apache поддержку различных языков программирования и систем разработки.

К ним относятся PHP, Python Ruby, Perl, ASP, Tcl.

Кроме того, Apache поддерживает механизмы CGI и FastCGI, что позволяет исполнять программы на практически всех языках программирования, в том числе C, C++, Lua, sh, Java.

Серверная часть приложения, реализованного в данной работе написана на языке программирования PHP, подключённого как модуль веб-сервера APACHE.

## PHP

PHP- это широко используемый язык сценариев общего назначения с открытым исходным кодом.

PHP - язык программирования, специально разработанный для написания web-приложений (скриптов, сценариев), исполняющихся на Web-сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и Perl. Он очень похож на С и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить веб-мастер, не знающий пока других языков программирования.

Огромным плюсом PHP, в отличие от, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. PHP не зависит от скорости компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML-файл или результат выполнения скрипта.

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в html страницы.

PHP способен генерировать и преобразовывать не только HTML документы, но и изображения разных форматов - JPEG, GIF, PNG, файлы PDF и FLASH. PHP способен формировать данные в любом текстовом формате, включая XHTML и XML.

PHP - кроссплатформенная технология. Дистрибутив PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix, Microsoft Windows, Mac OS и многих других.

## MYSQL

Информация, которую собирает и обрабатывает разработанное приложение хранится в базе данных. База данных - это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для динамических сайтов со значительными объемами данных — часто это интернет-магазины, порталы, корпоративные сайты.

В контексте баз данных стоит рассмотреть понятие СУБД. Система управления базами данных - это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL - оптимальный вариант для создания интернет-сайтов различной сложности.

MySQL - свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

## HTML, CSS, JS

Отправив запрос на веб-веб сервер пользователь зачастую получает в ответ документ, содержащий html, css и java script код.

*HTML*- это язык разметки гипертекста. Данный язык применяется для создания веб-страниц. Он интерпретируется браузером и отображается в виде документа в удобной для человека форме. HTML – это неотъемлемая составляющая и основа практически любой веб-страницы. Язык HTML в первую очередь выступает как средство логической разметки страницы. Именно HTML позволяет нам наделять содержимое страницы определенным смыслом, а реализуется это с помощью так называемых тэгов. Тэги – это специальные маркеры, которые определенным образом интерпретируются браузером. Суть тэгов в том, что содержимое страницы, заключенное в разные тэги, по-разному обрабатывается браузером. Язык HTML имеет достаточно длительную историю развития, и за это время претерпел значительные изменения. Большая часть изменений связана с добавлением в язык новых тэгов и «выводом из обращения» устаревших. На момент написания этой работы уже начата разработка версии HTML 5.1. Также следует упомянуть о том, что существует еще и язык XHTML. Это расширяемый язык разметки. На данный момент его развитие остановлено и рекомендуется использовать HTML. Новые версии XHTML не выпускаются. В заключение хочется отметить также тот факт, что средствами HTML все же можно управлять не только логической структурой веб-страниц, но и некоторыми аспектами ее внешнего вида. Однако такой подход считается неверным и устаревшим, т.к. для задания внешнего вида существует отдельный язык- CSS.

*CSS* – это язык описания внешнего вида документа, написанного с использованием языка разметки. Название произошло от англ. Cascading Style Sheets — каскадные таблицы стилей. Проще говоря, язык CSS предназначен для того, чтобы придавать необходимый внешний вид HTML-документам. Придание внешнего вида документам HTML – это хоть и самый популярный, однако лишь частный случай применения языка CSS, т.к. с его помощью можно придавать вид и документам других типов: XHTML, SVG и XUL. Раздельное описание логической структуры и представления документа позволяет более гибко управлять внешним видом документа и минимизировать объем повторяющегося кода, который бы неизбежно возникал при использовании HTML для описания внешнего вида документа. С помощью CSS веб-разработчик может задавать для страницы и отдельных ее элементов различные гарнитуры и размеры шрифта, цвета элементов, отступы элементов друг от друга, расположение отдельных блоков на странице и т.д. Для того, чтобы использовать CSS для придания внешнего вида HTML-документу, нужно этот документ связать со стилями. Для этого существуют различные способы подключения CSS к документу, которые дают браузеру знать, что к странице в целом, либо к каким-то отдельным ее элементам должно быть применено стилевое оформление. Таблицы стилей могут располагаться как непосредственно внутри того, документа, к которым они будут применяться, так и находиться в отдельном файле, имеющем расширение .css. Стили CSS могут быть подключены к странице четырьмя следующими способами:

* Если таблица стилей находится в отдельном файле, то она подключается к документу с помощью специального тэга link, который должен располагаться в этом документе внутри тэга head:

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

В атрибуте href при этом указывается путь до подключаемого файла стилей.

* Второй способ подключения применяется также в том случае, если стили находятся в отдельном файле. При этом используется директива @import, которая должна находиться в этом документе внутри тэгов style (которые, в свою очередь, должны находиться внутри тэгов head):

<head>

<style type="text/css" media="all">

@import url(style.css);

</style>

</head>

* Третий способ используется в случае, когда стили CSS расположены внутри того документа, к которому они должны применяться. В этом случае стили должны находиться в этом документе внутри тэгов style (которые, в свою очередь, должны находиться внутри тэгов head):

<head>

<style type="text/css">

body {

color: red;

}

</style>

</head>

* Четвертый способ также применяется в случаях, когда таблица стилей описана в самом документе. При этом стилевое оформление задается для какого-то конкретного элемента веб-страницы посредством использования его атрибута style.

При этом все CSS-правила, заданные таким образом, будут применяться только к текущему тэгу.

<p style="font-size: 21px; color: green;">Используем встроенные таблицы стилей.</p>

В процессе своего развития язык CSS уже прошел достаточно длинный путь, и в настоящее время существует несколько его уровней: CSS1, CSS2, CSS2.1, CSS3, CSS4.

*JAVA SCRIPT* изначально создавался для того, чтобы сделать web-странички «живыми». Программы на этом языке называются скриптами. В браузере они подключаются напрямую к HTML и, как только загружается страничка – тут же выполняются.

Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется.

Что же касается остальных возможностей – они зависят от окружения, в котором запущен JavaScript. В браузере JavaScript умеет делать всё, что относится к манипуляции со страницей, взаимодействию с посетителем и, в какой-то мере, с сервером: Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.

Реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п. Получать и устанавливать cookie, запрашивать данные, выводить сообщения…

Также JAVA SCRIPT позволяет реализовывать технологию AJAX.

## AJAX

AJAX — это аббревиатура, которая означает Asynchronous Javascript and XML. На самом деле, AJAX не является новой технологией, так как и Javascript, и XML существуют уже довольно продолжительное время, а AJAX- это синтез обозначенных технологий. При использовании AJAX нет необходимости обновлять каждый раз всю страницу, так как обновляется только ее конкретная часть. Это намного удобнее, так как не приходится долго ждать, и экономит трафик.

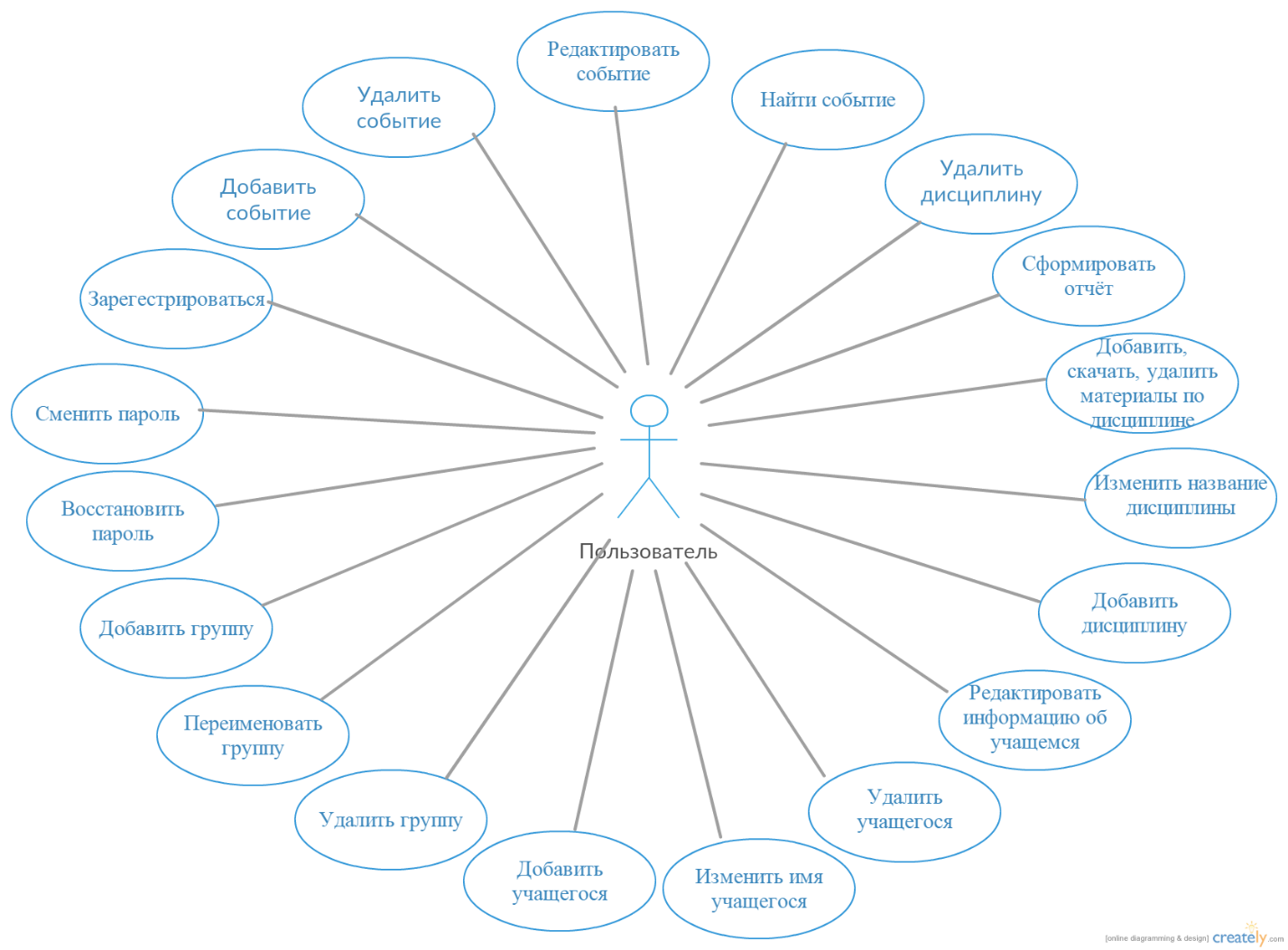
В данной работе для многих целей, в том числе и для использования AJAX используется популярная JS библиотека JQUERY, позволяющая сократить трудозатраты за счёт наличия удобных методов.

## заключение

# Разработка мобильного приложения

## Описание функциональности системы

Описание приложения начнём с рассмотрения диаграммы прецедентов. Диаграмма прецедентов в UML - диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой.



Опишем правильные сценарии работы с точки зрения пользователя, без учёта альтернативных вариантов:

**Название прецедента**: *Зарегистрироваться*

**Цель**: добавить пользователя в систему, установить для него пароль

**Предусловия**: пользователь зашёл на сайт и видит окно ввода логина и пароля

**Главная последовательность:**

1. пользователь выбирает пункт «регистрация»
2. система показывает окно ввода email и пароля
3. пользователь вводит желаемый email и пароль, нажимает кнопку зарегистрироваться
4. система отсылает электронное письмо с кодом подтверждения на введённый email.
5. Пользователь вводит код подтверждения в окно, предложенное системой

**Название прецедента**: *Сменить пароль*

**Цель**: изменить пароль пользователя

**Предусловия**: пользователь вошёл в систему, введя логин и пароль

**Главная последовательность:**

1. пользователь выбирает пункт «настройки» боковой панели
2. пользователь вводит старый и новый пароль, выбирает пункт «сменить пароль»

**Название прецедента**: *Восстановить пароль*

**Цель**:  восстановить забытый пароль

**Предусловия**: пользователь зарегистрирован в системе, зашёл на сайт и видит окно ввода логина и пароля

**Главная последовательность:**

1. пользователь выбирает пункт «восстановить пароль»
2. пользователь вводит email
3. пользователь вводит код восстановления, пришедший на email в окно, предложенное системой
4. пользователь получает новый, автоматически сгенерированный, пароль на email

**Название прецедента**: *Добавить группу*

**Цель**:  добавить группу в систему

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает пункт «новая группа»
3. пользователь вводит название новой группы и наживает «ок»

**Название прецедента**: *Переименовать группу*

**Цель**:  изменить название существующей группы

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает контекстное меню существующей группы
3. пользователь выбирает пункт «переименовать»
4. пользователь вводит новое название группы и наживает «ок»

**Название прецедента**: *Удалить группу*

**Цель**:  удалить существующую группу

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает контекстное меню существующей группы
3. пользователь выбирает пункт «удалить»
4. пользователь нажимает «ок» в диалоговом окне

**Название прецедента**: *Добавить учащегося*

**Цель**:  добавить нового учащегося

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает группу в которую необходимо добавить учащегося
3. пользователь выбирает пункт «новый ученик»
4. пользователь вводит ФИО нового ученика, нажимает «ок»

**Название прецедента**: *Изменить имя учащегося*

**Цель**:  изменить ФИО учащегося

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает контекстное меню необходимого учащегося
3. пользователь выбирает пункт контекстного меню «изменить»
4. пользователь вводит новое ФИО и нажимает «ок»

**Название прецедента**: *Удалить учащегося*

**Цель**:  удалить существующую запись об ученике

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает контекстное меню необходимого учащегося
3. пользователь выбирает пункт контекстного меню «удалить»
4. пользователь нажимает «ок»

**Название прецедента**: *Редактировать информацию учащегося*

**Цель**:  изменить информацию об учащемся, его фото.

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «ученики» боковой панели
2. пользователь выбирает необходимого ему учащегося
3. пользователь вводит необходимую информацию в поле текстового ввода и нажимает «запомнить», либо добавляет или удаляет фотографию, воспользовавшись контекстным меню фото.

**Название прецедента**: *Добавить дисциплину*

**Цель**:  добавить новую дисциплину в систему

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «дисциплины» боковой панели
2. пользователь выбирает пункт «новая дисциплина»
3. пользователь вводит название новой дисциплины, нажимает «ок»

**Название прецедента**: *Изменить название дисциплины*

**Цель**:  изменить название существующей дисциплины

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «дисциплины» боковой панели
2. пользователь выбирает пункт контекстного меню «изменить», необходимой дисциплины
3. пользователь вводит новое название дисциплины, нажимает «ок»

**Название прецедента**: *Добавить, удалить, скачать материалы по дисциплине*

**Цель**:  управлять материалами дисциплины

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «дисциплины» боковой панели
2. пользователь выбирает необходимую ему дисциплину
3. пользователь управляет материалами дисциплины, пользуясь пунктом «добавить материалы» и контекстным меню пункта списка материалов

**Название прецедента**: *Сформировать отчёт*

**Цель**:  Сформировать отчёт по необходимым пользователю параметрам

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «отчёты» боковой панели
2. пользователь выбирает тип занятия для формирования отчёта
3. пользователь выбирает тип отчёта
4. пользователь выбирает отчётную группу
5. при необходимости отчёта по конкретному учащемуся, пользователь выбирает учащегося
6. пользователь выбирает дисциплину
7. пользователь выбирает начальную и конечную дату
8. пользователь нажимает «сформировать»

**Название прецедента**: *Удалить дисциплину*

**Цель**:  удалить дисциплину из системы

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «дисциплины» боковой панели
2. пользователь выбирает пункт контекстного меню «удалить», необходимой дисциплины
3. пользователь положительно отвечает в появившимся диалоговом окне

**Название прецедента**: *Найти событие*

**Цель**:  найти необходимое пользователю событие

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «события» боковой панели
2. пользователь выбирает необходимые ему пункты, для сортировки событий
3. пользователь нажимает «найти»

**Название прецедента**: *Редактировать событие*

**Цель**:  редактировать информацию события

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «события» боковой панели
2. пользователь с помощью пункта найти события находит необходимые события
3. пользователь в контекстном меню необходимого ему события выбирает «изменить»
4. пользователь вносит необходимые изменения в информацию события

**Название прецедента**: *Удалить событие*

**Цель**:  удалить необходимое событие

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «события» боковой панели
2. пользователь с помощью пункта найти события находит необходимые события
3. пользователь в контекстном меню необходимого ему события выбирает «удалить»
4. пользователь положительно реагирует на вопрос системе о необходимости удаления

**Название прецедента**: *Добавить событие*

**Цель**:  добавить новое событие

**Предусловия**: пользователь вошёл в систему

**Главная последовательность:**

1. пользователь выбирает пункт «события» боковой панели
2. пользователь выбирает все пункты добавления нового события и при необходимости вводит необходимую дополнительную информацию
3. пользователь нажимает «добавить»

## Схема базы данных

Как уже было сказано выше, данные приложения хранятся в базе данных MySql. Каждый новый пользователь системы получает свой собственный набор таблиц, начинающийся с email пользователя. Например: пользователь с email [user@domain.what](mailto:user@domain.what) будет иметь таблицы с именами составленными по принципу [user@domain.what](mailto:user@domain.what)\_название-таблицы Это позволяет поддерживать оригинальность названий таблиц.

Система оперирует 1 общей таблицей для хранения учётных данных пользователей и 7 таблицами для каждого отдельного пользователя.

Таблица users может быть представлена схемой:

Опишем базу данных используя Схема базы данных— её структура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице (обычно с указанием их названия, типа, обязательности), и ограничения целостности (первичный, потенциальные и внешние ключи и другие ограничения).

Заключение

Список литературы

<https://pro-prof.com/archives/2594>

https://creately.com/app/?tempID=h165rwt81&login\_type=demo#