**Цель создания и функции**

Как и в любом проекте, главным вопросом является цель его создания. Какую пользу он принесет людям?

Данный проект разработан для того, чтобы процесс обучения в университете стал легче и удобнее для студентов, а так же для преподавателей.

Первым и главным преимуществом станет возможность дистанционного доступа студента к информации, касающейся процесса обучения. Это включает в себя информацию о предметах в группе студента, списке заданий и расписании занятий, оценок и посещаемости, а также контактных данных преподавателей, расписании их консультаций и основной информации о структуре университета. Благодаря этому студенту не придется тратить время на ненужные вещи, и он сможет сохранить время и силы для обучения.

Вторым преимуществом является то, что данные студента останутся конфиденциальными. Никто кроме самого студента и преподавателя не сможет узнать его оценки и посещаемость.

**Безопасность и надежность**

К обеспечению безопасности данных можно отнести:

* Хранение паролей в хэшированом виде.
* Наличие системы прав доступа. Каждый пользователь имеет доступ лишь к тем данным, которые ему доступны. У студента это его личные данные, данные о его группах: статус в ней, платежи, задания и оценки, занятия и посещаемость, а так же контактные данные преподавателей, расписание их консультаций и общие сведения о структуре университета. У пользователей доступ настраивается с помощью прав. В основном существует по одному праву на раздел. Также можно выбрать степень доступа: нет, только просмотр, просмотр и редактирование.  
  При попытке получить запрещенную информацию пользователь увидит соответствующее сообщение об ошибке.

К надежности можно отнести обеспечения правильности введенных пользователем данных за счет:

* Проверки правильности введенных в поля данных. Если есть ошибки, то информация не сохраниться и пользователь увидит сообщение об ошибке.
* Подтверждение выполнения действий (например, удаления) для того чтобы удостовериться, что это не было сделано случайно.
* Проверка возможности выполнения действий. Если данные, например, нельзя удалить по какой-либо причине, то это действие будет запрещено.

**Методы разработки**

Программа была разработана в виде веб-сайта.

В основном, такой выбор был сделан за счет относительно простоты разработки клиент-серверного взаимодействия и удобного пользовательского интерфейса. А так же благодаря этому программой можно воспользоваться, имея при себе просто телефон с выходом в интернет.

Для разработки интерфейса использовались HTML5 и CSS3.

Для клиентской части – JavaScript.

Для серверной части – PHP.

Также при разработке использовалась парадигма ООП и шаблон MVC.

**Разработка БД**

Разработке базы данных в этом проекте уделялось больше всего внимания. Так как в данном случае это самое главное. Потому как главная цель этой системы – быстро получить правильные данные.

Для этого было реализовано следующее:

* Нормализованная БД. Нет лишних данных, все, что можно, вынесено в отдельные таблицы, связано FK. Минимизирована возможность внести неверную информацию. Так же к этому можно отнести и проверку вводимых пользователем данных.
* Все запросы вынесены в методы классов и протестированы. Так как они находятся в отдельном методе, можно вносить изменения в одном месте, и они сразу произойдут во всех местах, где это используется.
* Для более быстрой обработки запросов, в них четко прописываются выходные данные, а так же для списков применяется пагинация, то есть постраничное разбиение. Таким образом, пользователь получает доступ к нужным данным максимально быстро.

**MVC + Smarty**

В проекте был реализован шаблон MVC (Model-View-Controller).

Суть его состоит в четком разграничении частей программы:

* Модель – отвечает за данные
* Вид – отвечает за отображение данных
* Контроллер – отвечает за взаимодействие всех компонентов системы и взаимодействие с пользователем

Таким образом, пользователь делает какое-то действие в системе. Вид передает информацию об этом в Контроллер. Контроллер при необходимость взаимодействует с Моделью (считывает или записывает какие-то данные в БД). Затем говорит Виду отобразить нужную информацию пользователю.

Для пользователя нет особой пользы от использования MVC. Польза для разработчика и для тех, кто будет потом поддерживать систему.

С MVC код программы становится более структурированным и понятным. Такую программу гораздо проще разрабатывать и в дальнейшем отлаживать. Благодаря такой схеме, становиться проще распознать в какой части программы ошибка: неправильные данные получены из Модели или Вид их не правильно интерпретировал или какие-то проблемы в Контроллере. Тем, кто будет поддерживать эту систему, будет гораздо проще разобраться с кодом такой программы и суметь в ней что-то исправить и добавить.

Реализован он при помощи шаблонизатора Smarty.

**ООП**

При разработке также использовалась парадигма ООП.

В системе достаточно простая система классов. В основном они используются для хранения функций взаимодействия с БД. Только у класса Auth большие обязанности, он отвечает за авторизацию пользователей и за предоставление им доступа к тем или иным данным.

**AJAX**

Так же стоит отметить, что в проекте используется технология AJAX, позволяющая асинхронно взаимодействовать с сервером.

Например, рассмотрим авторизацию пользователя. По сути, пользователь должен заполнить 2 поля: Email и пароль. Затем сервер проверит правильность и выдаст результат (можно войти или нет).

Без AJAX это будет выглядеть так: пользователь введет данные (например, неправильные), нажмет кнопку входа, отправится запрос на сервер, он обработает эту информацию и потом заново сгенерирует страницу уже с сообщением об ошибке и полностью отправит ее клиенту. У пользователя в это время будет перезагружаться страница и через некоторое время он увидит обновленную страницу с сообщением об ошибке. Так же при этом могут пропасть введенные данные.

С AJAX при вводе данных и нажатии на кнопку на сервер асинхронно (пользователь этого не увидит) отправится просто 2 переменных, он их обработает и сразу вернет результат. Далее сам клиент на основе полученных данных либо запустит пользователя в систему, либо выведет сообщение об ошибке. Тем временем страница в браузере пользователя не будет перезагружаться, и он практически мгновенно увидит сообщение об ошибке.

То есть AJAX очень полезен как серверу, так и пользователю:

* Передается меньше лишней информации
* Страница лишний раз не обновляется и это выглядит приятнее для пользователя

**Поддержка проекта**

Поддерживать проект достаточно просто за счет того, что при разработке использовались известные и современные технологии. А так же благодаря простой и удобной структуре программы, сделанной за счет ООП и MVC. Так же плюсом есть достаточно простая, нормализованная БД.

**Развитие проекта**

Кроме багфиксов, конечно, можно дополнять систему различными новыми функциями. Например:

* Оповещения в виде Email и SMS при различных событиях в системе
* Разработка мобильной версии для удобства использования
* Отчеты в формате PDF и Excel
* Календарь событий
* Форум для общения и передачи файлов
* Хранение документов и электронная подпись
* Проведение финансовых операций