**Процесс создания, актуализации и согласования РПД**

Из года в год преподаватели и профессора вузов занимаются созданием и актуализацией рабочих программ дисциплин, которые играют важную роль в образовательном процессе.

Рабочая программа дисциплины представляет собой нормативный документ, в котором определены объем знаний, содержание материала, порядок изучения и преподавания учебной дисциплины. Данный документ устанавливает методы контроля и оценки результатов освоения дисциплины, учитывает специфику подготовки студентов по выбранному направлению, обеспечивая качественное образование и развитие необходимых навыков и компетенций, а так же соответствующие требованиям ФГОС ВПО (Федерального государственного образовательного стандарта высшего профессионального образования) для соответствующих направлений подготовки.

Содержание и структура рабочих программ дисциплин (РПД) утверждаются методическим советом университета. Данные для составления РПД берется частями, одна из учебного плана направления и основной профессиональной образовательной программы (ОПОП), а самую содержательную часть описывает преподаватель, ответственный за данную дисциплину.

Актуализация РПД происходит ежегодно и может быть обусловлена различными факторами, включая:

* утверждение нового образовательного стандарта;
* изменения в ОПОП;
* изменения в структуре РПД;
* необходимость обновления тем и содержания дисциплины;
* обновление материально-технического обеспечения (МТО).

Структура РПД, а также источники данных, используемые при ее формировании, представлены в таблице 1.

Табл. 1. Структура РПД

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № раздела | Раздел РПД | Данные из учебного плана | Заполняет преподаватель | Доп. документы |
| 1 | Титульный лист |  |  | + |
| 2 | Лист согласования |  |  | + |
| 3 | Цели и задачи освоения дисциплины |  | + |  |
| 4 | Место дисциплины в структуре ОПОП | + |  |  |
| 5 | Планируемые результаты обучения по дисциплине (модулю) | + |  |  |
| 6 | Объем дисциплины | + |  |  |
| 7 | Содержание дисциплины |  | + |  |
| 8 | Учебно-методическое обеспечение |  | + |  |
| 9 | Необходимое материально-техническое обеспечение |  | + |  |

### 

## **Конструктор образовательных программ ИТМО**

<https://op.itmo.ru/> Один из наиболее успешных примеров комплексного решения в области разработки образовательных программ — Конструктор образовательных программ ИТМО. Этот инструмент предоставляет возможность быстрого создания, сопровождения и планирования образовательных программ, охватывая все необходимые элементы РПД.

С использованием данного сервиса преподаватели могут в удобном формате просматривать все имеющиеся рабочие программы, а также статус их разработки (см. рис. 1). Однако, что делает его особенно эффективным, это наличие встроенной валидации данных, которая помогает преподавателям быстро и точно заполнять рабочую программу дисциплины, избегая ошибок и экономя время.

Изображение выглядит как снимок экрана, текст, число

Автоматически созданное описание

Рисунок 1. Работа с РПД

Конструктор образовательных программ ИТМО представляет собой не только простой инструмент для создания РПД, но и полноценный методический конструктор. Он включает в себя функционал работы с учебными планами (см. рис. 2), общими характеристиками образовательных программ, навигатором по онлайн-курсам и возможностью генерации отчетов. Все вместе это обширный функционал, который позволяет преподавателям эффективно управлять всем процессом образования.

Изображение выглядит как текст, число, снимок экрана, чек

Автоматически созданное описание

Рисунок 2. Работа с учебными планами

Важно отметить, что Конструктор образовательных программ ИТМО полностью адаптирован под специфические процессы и требования ИТМО. Это может создавать некоторые трудности при его применении в других университетах, где используются отличные от ИТМО бизнес-процессы. Однако, даже само наличие подобного комплексного решения может быть ценно для других учебных заведений при создании собственных инструментов, учитывающих их особенности и потребности.

## **Требования к функционалу приложения**

* Администратор: Создание рабочих программ дисциплин на новый учебный год на других РПД прошлого учебного года или данных, полученных из учебных планов. Данные из учебного плана, необходимые для создания РПД приведены в таблице 1.
* Преподаватель: Редактирование и обновление рабочих программ доступные только ему, включая изменение содержания, добавление новых разделов. Данные в РПД, полученные из учебного плана для преподавателя не доступны к редактированию.

Изображение выглядит как диаграмма, текст, линия, План

Автоматически созданное описание

Рисунок 5. Диаграмма нулевого уровня

В нашей системе, *UML* диаграмма use-case (см. рис. 8-9) используются для более полного понимания и визуализации функциональности и взаимодействия различных компонентов и актеров в рамках конструктора рабочих программ.

Изображение выглядит как текст, диаграмма, снимок экрана, круг

Автоматически созданное описание

Рисунок 8. *Use-case* диаграмма

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 9. *Use-case* диаграмма

В рамках серверной части конструктора рабочих программ дисциплин преподаватель имеет возможность:

* авторизация в системе;
* редактирование существующей рабочей программы: Преподаватель может открыть существующую рабочую программу для внесения изменений. Он может изменить те разделы, которые не пришли от сервера из учебного плана;
* просмотр доступной информации: Преподаватель может запрашивать информацию о существующих рабочих программах;

Администратор имеет возможность:

* создание новых рабочих программы: администратор запускает конструктор РПД и тот в свою очередь создает для всех пользователей РПД. После чего если необходимо, он может изменить данные в созданном РПД или удалить его;
* создание пользователей с ролью преподаватель. После чего если необходимо, он может изменить данные пользователя или удалить его;
* создание новых связей между дисциплинами и преподавателями. После чего если необходимо, он может изменить данные в созданной связи или удалить ее;
* просмотр доступной информации: администратор может запрашивать информацию о существующих рабочих программах;
* загрузка файла учебного плана в программу и если необходимо удалить его;

При создании новой рабочей программы дисциплины (РПД) на основе данных из учебного плана, приложение взаимодействует с сервером и получает все необходимые данные, которые должны быть включены в РПД на основе учебного плана (см. таблицу 1).

Этот процесс включает следующие шаги:

* Приложение обращается к серверу и запрашивает данные из учебного плана для конкретной дисциплины.
* Сервер обрабатывает запрос и отправляет приложению соответствующие данные из учебного плана.
* Приложение принимает полученные данные и анализирует их содержимое.
* На основе этих данных приложение автоматически заполняет соответствующие разделы и поля в новой РПД.
* После завершения процесса создания РПД, приложение сохраняет полученные данные и предоставляет возможность просмотра, редактирования и экспорта готовой РПД для дальнейшего использования.

Таким образом, приложение на основе данных из учебного плана автоматически заполняет соответствующие разделы и поля новой РПД, упрощая процесс создания программы и обеспечивая ее актуальность и соответствие учебному плану.

В результате этапа проектирования была получена ясная представление о функциональности и взаимодействии компонентов в серверной части конструктора рабочих программ. Данная информация будет служить основой для разработки и реализации конечного продукта.

## **Физическая модель данных**

Физическая модель базы данных — это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных (см. рис.10 ).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 10. *Логическая модель данных*

Данная логическая модель содержит 8 сущностей:

* сущность Documents хранит в себе информацию об учебных планах, а именно название (name) и год (year);
* сущность Detail хранит детальную информацию о дисциплине, имя, код, семестр, курс и тд;
* сущность Descipline хранит краткую информацию о дисциплине;
* сущность Users хранит информацию о преподавателях , а именно ФИО и логин;
* сущность Curs\_Semestr, хранит связь между таблицей курсом, семестром и дисциплиной;
* сущность Desp\_Users\_Con, хранит связь между преподавателями и дисциплинами;
* сущность listSoglTable, хранит в себе данные для листа согласования;
* сущность Rpd, хранит сгенерированные системой РПД;

# **Реализация серверной части информационной системы**

## **Архитектура веб-приложения конструктор рабочих программ дисциплин**

Система «Конструктор рабочих программ дисциплин» построена на основе *REST* API и позволяет использовать концепцию *REST* в работе с клиентской частью програмы.

*REST* — это архитектура для организации взаимодействия между независимыми приложениями посредством протокола *HTTP* и включает в себя набор рекомендаций взаимодействия клиент-серверных приложений.

Каждый объект на сервере имеет свой уникальный URL-адрес в строгом последовательном формате и чтобы им воспользоваться в REST API есть 4 метода HTTP, которые используют для действий с объектами на серверах(см. рис. 11):

* GET (получение информации о данных или списка объектов)
* DELETE (удаление данных)
* POST (безопасное добавление или замена данных)
* PUT (регулярное обновление данных)

Изображение выглядит как снимок экрана, диаграмма, текст, линия

Автоматически созданное описание

Рисунок 11. *REST API* диаграмма

## **Архитектура серверной части приложения**

Для серверной части приложения используется паттерн программирования *MVC.* Основная цель применения этой концепции состоит в отделении бизнес-логики от её представления. За счёт такого разделения повышается возможность повторного использования кода.

 Схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

В серверной части «Конструктора рабочих программ дисциплин» как было сказано ранее база данных, документы РПД и дисциплины - это модели, так как содержат данные, которые мы вынесем в специальное место для их хранения и назовем его *Models*. Чтобы показать данные, мы создадим специальные компоненты позволяющие выводить информацию о них и контроллеры – специальную прослойку между видом и моделью, которая ждет запроса на выполнение вывода информации, в нашей работе и будет иметь имя *inc*.

Из-за чего файловая структура проекта примет такой вид (см. рис. 12)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 12. *Файловая структура проекта*

В папке *Models* лежат 5 моделей необходимых для работы приложения:

* *Database.php*: (модель подключения к базе данных)
* *Descipline.php*: (модель дисциплины)
* *Document.php*: (модель документа учебного плана)
* *PostsModel.php*: (модель позволяющая делать запросы к базе данных)
* *Rpd.php*: (модель рабочей программы десциплины)

Все вышеописанные классы описывают модель классов (см. рис. 13)

Изображение выглядит как текст, диаграмма, План, схематичный

Автоматически созданное описание

Рисунок 13. *Модель классов*

### ***Database.php***

В системе «Конструктор рабочих программ дисциплин» файл с именем Database.php необходим для обеспечения подключения программы к базе данных mySQL (см. приложение 1).

Логика класса проста, в блоке try-catch производится попытка соединения с базой данных с помощью PDO метода setAttribute();. Данные для подключения, а именно поля hostname, username, password и *DB\_NAME* берутся из другого файла в проекте, consfig.php (см. рис. 14). Это специальный файл с отдельно вынесенными полями и установленными как не изменяемые, сделано из соображения безопасности и идеологии MVC.

Select() — функции, созданнаяа из соображений экономии времени и необходима в будущем. Для этого человек будет делать запрос через эту функцию.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 14. *Поля для подключения, вынесенные в отельный файл config.php*

### ***PostsModel.php***

Самый большой класс по объёму строк в системе «Конструктор рабочих программ дисциплин» файл с именем *PostsModel*.php наследуемый от предыдущего файла *Database*.php создан с целью обеспечения запросов в базу данных.

Разделенный визуального на блоки кода этот файл обладает следующими методами (реализация методов приведена в приложение 2):

* Блок для работы с учебным планом:
  + метод дающий список всех дисциплин и их полей;
  + метод, заносящий все поля дисциплины;
  + метод, дающий детальный просмотр дисциплины;
  + метод дающий максимальный элемент в таблице Documents;
  + метод дающий максимальный элемент в таблице Detail;
  + метод, заносящий все дисциплины в базу данных;
  + метод, заносящий все документы учебного плана в базу данных;
  + метод выдающий список дисциплин;
  + метод выдающий список документов;
  + метод выдает одну дисциплину;
  + метод, удаляющий документы из базы данных;
* Блок пользователей:
  + метод добавляющий в пользователя в базу данных;
  + метод выдает в пользователя из базы данных;
  + метод изменяет пользователя в базе данных;
  + метод удаляет пользователя из базы данных;
  + метод создающий связь между пользователем и дисциплиной в базу данных;
  + метод выдает список связей между пользователем и дисциплиной в базе данных;
  + метод обновляет список связей между пользователем и дисциплиной в базе данных;
  + метод удаляет связь между пользователем и дисциплиной в базе данных;
* Блок РПД:
  + метод создающий и добавляющий РПД в базу данныз;
  + метод выдает список созданных РПД;
  + метод выдает РПД только указанного пользователя;
  + метод удаляющий РПД из базы данных;
  + метод изменяющий РПД в базе данных;

### ***Documents.php***

Класс, работающий с учебным планом. После загрузки в систему документа, он конвертируется, преобразовывается и проходит процесс извлечения информации с последующей за этим загрузкой в базу данных(см. приложение 3).

Метод *upload()* – загружает документ в систему в папку *uploads* (см. рис. 12)

Метод *deleteString()* – проводит первичную очистку файла от ненужных полей и готовит его конвертации в другой формат

Метод *toJson* () – производит конвертацию из .*plx* формата в формат .*json* (см. рис. 36)

Ключевой метод в данном классе *insertData()* – он загружает все данные из учебного плана в базу данных

### ***Descipline.php***

Класс нужен для формирования объекта дисциплина. участвующего в работе всей системы

Объект генерируется методом – *all*()

### ***RPD.php***

Класс нужен для формирования объекта RPD, участвующего в работе всей системы.

## **Логика работы серверной части**

Суть приложения - предоставить преподавателю уже собранный РПД в конструкторе и дать возможность дополнить его. Все вышеописанные классы, по сути, нужны для единственной вещи – создать РПД.

Чтобы создать РПД, необходимо взять данные из учебного плана, информацию о связи с преподавателем и дисциплиной и добавить дополнительные поля листа согласования.

Логика приложения такова, что при загрузки учебного плана он преобразуется и из него извлекаются данные о дисциплинах, которые сразу попадают в базу данных и ждут дальнейшего использования. (см. рис. 15)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15. *Логика загрузки файла ч1*

При использовании *POST* запроса *uploadFile,* отрабатывают 4 метода описанные выше. Файл загружается, из него вырезаются ненужные строки, файл конвертируется в jsonи как было сказано ранее происходит запись данных в базу данных.

Тем временем администратор заполняет информацию о преподавателях и настраивает связи преподавателей с дисциплинами которые они преподают с помощью запроса addConnection() (см. рис. 16)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 16. *Логика загрузки файла ч2*

API функция addInList() нужна для заполнения информации о листе согласования.

При использовании *POST* запроса *addInUser и addConnection,* администратор может заполнить данные в таблицу о пользователя и связи его с предметами.

После всего можно воспользоваться *POST* запроса createRpd и сзодать для всех пользователей РПД. (см. рис. 17)

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 17. *Логика загрузки файла ч3*

После того как система создала РПД преподаватель может зайти и посмотреть доступные для него дисциплины. Для демонстрации я буду пользоваться программой PostMan.

PostMan - платформа API, позволяющая разработчикам проектировать, создавать, тестировать и повторять свои API.

GET запрос для предоставления данных которым мы воспользуемся для показа пользователя – <https://konstr:8890/api/getAllRpd> (см. рис. 18)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 18. *Показ РПД*

Все операции, которые была описаны, выше позволяют создать РПД. Это основная функция приложения и ее типовой сценарий. Однако в приложении две роли: пользователь и администратор и следует описать и продемонстрировать компоненты для работы с ними детально.

Так преподаватель, как было описано ранее умеет и должен вносить данные в РПД. Это можно будет сделать, с помощью PATCH запроса и API ссылки https://konstr:8890/api/updateRpd/$id (см. рис. 19) где $id – id РПД в системе.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 19. *Обновление данных РПД*

Также преподаватель, может получить список своих доступных ему РПД, с помощью GET запроса и API ссылки <https://konstr:8890/api/getAllRpd> (см. рис. 20)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 20. *Список доступных данных пользователей*

Администратор может загрузить файл учебного плана POST запросом и ссылкой

<https://konstr:8890/api/uploadFile> (см. рис. 21)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 21. *Команда на загрузку учебного плана в приложение*

Администратор может удалить файл учебного плана POST запросом и ссылкой

<https://konstr:8890/api/uploadFile>(см. рис. 22)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 22. *Команда на удаление документа учебного плана*

Таким образом в приложении каждая таблица обладает CRUD методом, реализованным через API ссылку. Но нюанс в том, что использовать все методы может только администратор системы, преподаватель может только получать РПД его специальности и заполнять некоторые данные разрешенные ему.

## **Интеграция с клиентской системой**

При первом входе в систему пользователь увидит перед собой страницу авторизации (см. рис. 23).

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 23. Страница авторизации

Если до этого, пользователь не был зарегистрирован, то при нажатии на кнопку «Зарегистрироваться», он перейдет к интерфейсу регистрации (см. рис. 24).

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 24. Страница регистрации

Как уже было сказано ранее, на обеих страничках настроена вариация, например при попытке ввода русских символов, пользователь сразу получит ошибку (см. рис. 25.).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 25. Ошибка валидации

После заполнения всех полей, пользователь попадает на страничку направлений (см. рис. 26).

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рисунок 26. Страница направлений

На данной странице отображаются все РПД, относящиеся к авторизованному преподавателю. Это позволяет преподавателю получить полный список доступных документов и ознакомиться с ними.

Чтобы отобразить данный список, нужно использовать *FETCH* запрос на стороне клиента с использованием GET запроса, на адрес <https://konstr:8890/api/getRpd/$id>, где $id это логин преподавателя. Fetch – механизм JavaScript который может отправлять сетевые запросы на сервер.

Если пользователь захочет записать какую то информацию он может редактировать запись и занести в нее свои данные. В свою очередь данные которые пришли с сервера – он не может редактировать.

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рисунок 27. Форма редактирования полей

Чтобы отобразить детальные данные, нужно использовать *FETCH* запрос на стороне клиента с использованием GET запроса, на адрес <https://konstr:8890/api/getDetailRpd/$id>, где $id это логин преподавателя.

После внесения данных, и нажатия на кнопку готово. Будет использован *FETCH* запрос на стороне клиента с использованием PATCH запроса, на адрес <https://konstr:8890/api/updateRpd/$id>, где $id это логин преподавателя. Данные обновятся и сохранятся в базе данных.

Таким образом, пользователь может подробно описать все необходимые ресурсы, которые могут потребоваться в процессе обучения по данной дисциплине. Это позволит создавать информативные и полноценные рабочие программы, которые содержат все необходимое обеспечение для успешного освоения дисциплины и достижения поставленных целей.

На данный момент осуществлена возможность непосредственно сгенерировать документ рабочей программы дисциплины из системы «Конструктора рабочих программ дисциплин». Однако, важно отметить, что вся система разработана с учетом будущей работы над проектом и его развитием. А это значит, что в будущем можно будет генерировать документы в различных форматах данных, таких как *PDF*, *DOC* и т.д.

Предусмотренный функционал позволяет собирать и хранить все необходимые данные, включая информацию о целях, задачах, компетенциях, обеспечении и других аспектах рабочей программы дисциплины. Это означает, что, данные с легкостью отправляются на сервер для формирования соответствующего документа.

Таким образом, все необходимые данные могут быть переданы на сервер для создания полноценного документа рабочей программы дисциплины, что позволит пользователям системы получить готовые и качественные документы, соответствующие требованиям и стандартам образовательных учреждений.