**Орлов Антон Сергеевич**

**Выпускная квалификационная работа бакалавра**

**Разработка прототипа системы управления центром развития электронных образовательных ресурсов**

Направление 01.03.02

Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,  
старший преподаватель,  
доцент  
Севрюков С. Ю.

Санкт-Петербург

2018

**Оглавление**

[**Глава 1. Введение** 2](#_Toc511645597)

[**2.1. Существующие реализации и альтернативы** 3](#_Toc511645598)

[**2.2. Вывод** 7](#_Toc511645607)

[**Глава 3. Проектирование** 8](#_Toc511645608)

[**3.1.** **Компоненты** 9](#_Toc511645609)

[**3.2.** **Процесс создания курса** 10](#_Toc511645610)

[**3.3.** **Процесс создания презентации** 11](#_Toc511645611)

[**3.4.** **Процесс создания видео** 13](#_Toc511645612)

[Разработка прототипа 14](#_Toc511645613)

[Тесты 14](#_Toc511645614)

# **Введение**

Центр развития электронных образовательных ресурсов (ЦРЭОР) Санкт-Петербургского государственного университета (СПбГУ) является организацией, создающей онлайн-курсы СПбГУ. Процесс создания курса достаточно сложный. В нем участвуют несколько человек:

* Автор - создает презентации, сценарии, задания и дополнительные материалы курса. Часто является преподавателем университета.
* Методист - поддерживает связь и взаимодействие с автором для согласования разрабатываемых материалов центром и оказывает помощь в разработке этих материалов автору. Просматривает все промежуточные результаты, выполняет поиск ошибок и неточностей. Оформляет курсы на образовательных платформах. Ведет мониторинг платформ, помогает слушателям и авторам решать возникающие вопросы от лица ЦРЭОР.
* Дизайнер - по исходным материалам, присланных автором, создает презентацию, оформленную по стандартам ЦРЭОР. Прорабатывает некоторые визуальные составляющие курса.
* Режиссёр - проводит съёмки видеоматериалов. Занимается обслуживанием студии и техники.
* Монтажер - выполняет монтаж видеороликов, соединяет снятые видеоматериалы со свёрстанными презентациями.

В процессе разработки курса создается множество материалов: презентации, видео, сценарии. Они проходят через несколько этапов редактирования участниками процесса. Из-за этого образуется много работ и файлов, состояние которых необходимо отслеживать. Для этого часто приходится связываться с другими сотрудниками, что замедляет процесс.

Для упрощения управлением процессом создания онлайн-курсов была разработана система, рассматриваемая в данной работе.

За время работы центра было отснято %%ГБ видео материалов, выпущено %% курсов, %% из которых размещаются на платформе Национальная платформа открытое образование (НПОО) и %% -- на Coursera. Курсы на НПОО имеют %%-%% модулей, а на Coursera %%-%%. Один модуль соответствует одной неделе обучения. Каждый модуль содержит одну презентацию, в которой примерно 50 слайдов, задания, дополнительные материалы и видео.

**Постановка задачи**

Целью работы является упрощение управления процессом создания онлайн-курсов руководством, работниками и авторами онлайн курсов. Задача состоит в разработке прототипа системы управления ЦРЭОР, который позволяет хранить материалы курса и данные персонала, отслеживать статусы работ, проводимых в центре и автором. Разработку прототипа необходимо вести с целью создания на его основе полноценной системы управления. В результате система должна стать SaaS приложением, которое поможет другим организациям создавать онлайн-курсы. Решение этой задачи помогает достичь цели. После того, как прототип будет создан, к нему будут добавляться элементы, такие как сервисы, интеграции с платформами, добавление модулей взаимодействия со студентами. Из-за этого разработку прототипа уже нужно вести так, чтобы он мог масштабироваться.

# **Глава 1. Существующие реализации и альтернативы**

В результате поиска существующих решений поставленной задачи были найдены некоторые реализации, которые решают часть проблем управления центром, но не покрывают все потребности и имеют совершенно другие приложения.

### iSpring <https://www.ispring.ru/?utm_source=meduza.io&utm_medium=referral&utm_campaign=cards>

Это решение позволяет создавать курсы, однако доступ к ним студенты получают через эту же платформу. Студенты курсов ЦРЭОР используют популярные MOOC, такие как “Открытое образование” и “Coursera”, так как они привычнее для них. ЦРЭОР заинтересован в выпуске своих курсов на популярных платформах, потому что это способствует расширению аудитории курсов.

Плюсы решения:

* Аналитика по курсам
* Хранение в облаке
* Самостоятельная платформа для обучения (придаёт гибкости)

Эти плюсы можно выделить из описания платформы на её официальном сайте. Теперь рассмотрим минусы этого решения по отношению к целям данной работы.

* Платформа не представляет инструментария для регулирования процесса создания сопроводительных материалов

Такое решение подходит индивидуальным создателям курсов и оно действительно позволяет упростить процесс. Однако, когда речь идёт о создании курса, представляющего университет, то необходимы большие возможности.

### Redmine

На данный момент используется только это приложение для регулирования процесса. Оно имеет ряд недостатков:

1. Отсутствует валидация ссылок, что приводит к ошибкам.
2. Нет интерфейса для работы с файлами.
3. Нет описания курса. Сотрудники пишут в поле для комментариев шаги, которые были проведены в процессе создания курса, однако для всех курсов они одинаковые. Этот процесс можно ускорить.

Поскольку приложение уже используется, а у сотрудники запрашивают другое решение, можно сделать вывод, что решение не удовлетворяет потребностям

Помимо существующих решений был сделан обзор альтернатив. Рассматривались Content Managment Systems (CMS), Digital Asset Management (DAM) системы и Video DAM.

### Webdam [*https://webdam.com/what-is-digital-asset-management/*](https://webdam.com/what-is-digital-asset-management/)

1. Хранение данных в облаке. Разные уровни доступа позволяют регулировать доступ к данным разных пользователей.
2. Организация и контроль: структура, ключевые слова и поиск.
3. Оповещения, комментирование, контроль версий и статус работы (aprrovals).
4. Контроль прав: лицензии, watermarks, соглашения и подобные метаданные.
5. Фильтры для изображений, поиск по документам.
6. Поддержка видео, аудио и изображений. Превью для этих типов данных. Конвертация видео файлов.
7. Инструменты для маркетинга: публикация в сторонние сервисы, взаимодействие с третьими лицами.

### Widen [*https://www.widen.com/what-is-digital-asset-management*](https://www.widen.com/what-is-digital-asset-management)

1. Организация данных индивидуально или группами.
2. Контроль листы и роли пользователей.
3. Поддержка разных типов файлов.
4. Эскизы
5. Введение метрик для отслеживания статуса выполнения работы.
6. Отношения между объектами.
7. Распределение данных на одних уровнях и централизация на других.
8. Поиск
9. Превью
10. Sharing, linking и другие способы выходить за пределы системы.

### Adobe Experience Manager [*http://www.adobe.com/ru/marketing-cloud/experience-manager/assets-digital-asset-management.html*](http://www.adobe.com/ru/marketing-cloud/experience-manager/assets-digital-asset-management.html)

1. Интеграция в облако.
2. Поиск и организация метаданных.
3. Проекты, задачи, потоки работ (workflow).
4. Медиа и работа с ними.
5. Умные теги и аналитика.
6. Работа с третьими лицами.

### HelloVideo (*https://codecanyon.net/item/hellovideo-video-cms/11149598)*

1. Добавление постов и страниц
2. Категории и меню
3. Статистика и аналитика
4. Работа с облаком

### Panopto ([*https://www.panopto.com/features/video-cms/video-content-management/*](https://www.panopto.com/features/video-cms/video-content-management/)*)*

1. Поддержка разных форматов
2. Контроль доступа
3. Загрузка видео drag&drop
4. Аналитика и отчеты
5. Возможность делиться видео
6. Поиск

### ASP.NET jVideo ([*https://www.mediasoftpro.com/product/beetube/index.html*](https://www.mediasoftpro.com/product/beetube/index.html) *)*

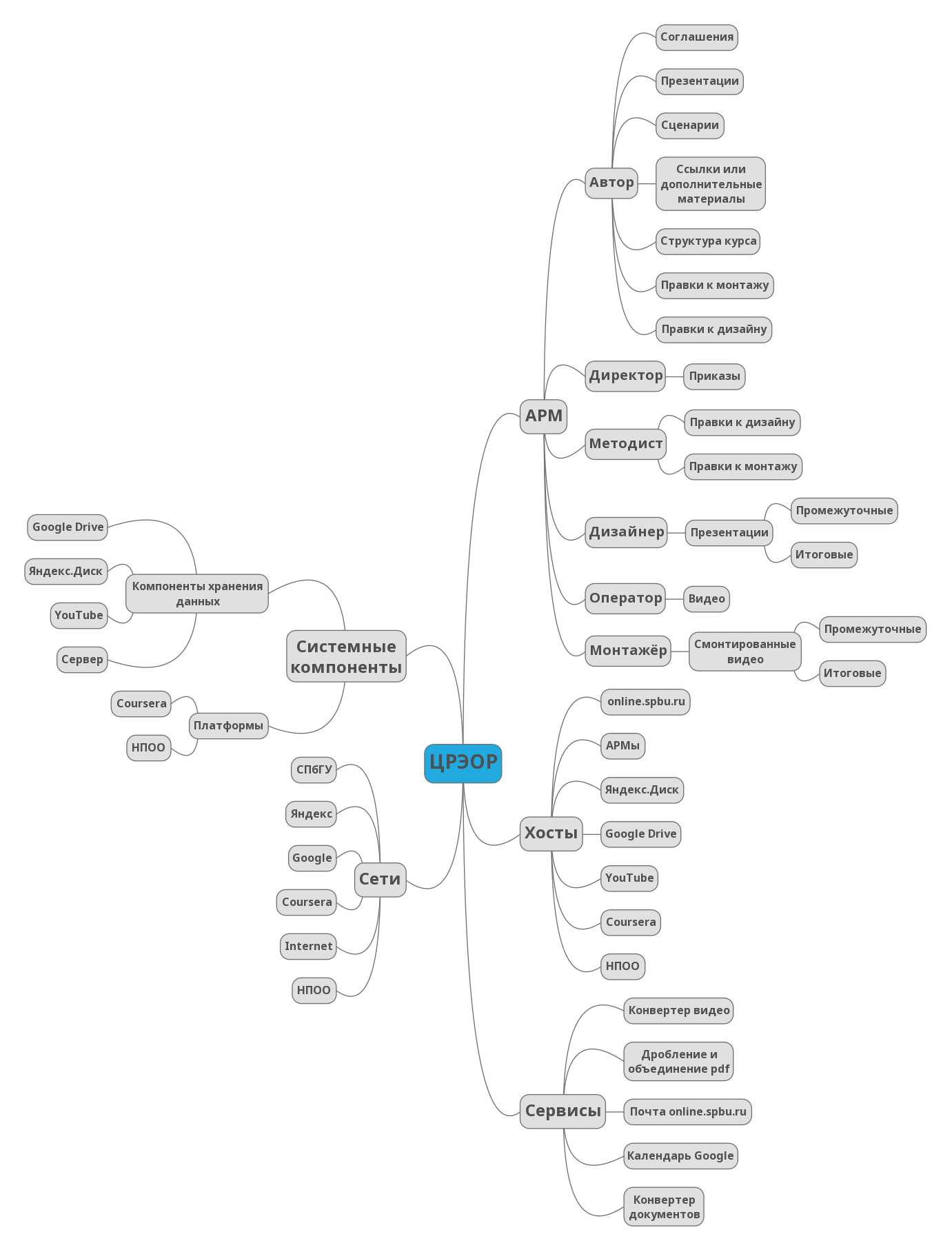
1. Альбомы
2. Блоги
3. Работа в облаке
4. Публикация медиа
5. Dashboard
6. Инструменты администрирования
7. Модули
8. Кастомизация
9. Фильтрация

## **2.2. Вывод**

В результате рассмотрения существующих решений и альтернатив выявляются элементы, которые присутствуют во всех системах подобного типа:

1. Структурированность. Assets должны иметь различные поля для более наглядного отображения. У элементов можно будет посмотреть подробности. Необходимо наличие полей с ключевыми словами для упрощения поиска. Заполнение полей для asset’а можно сделать автоматическим. Например, теги могут создаваться исходя из заголовка и содержимого каждого элемента.
2. Поиск. Так как количество элементов будет постоянно расти, со временем их количество станет таким, что невозможно будет пробежаться взглядом и выявить нужные элементы. Поиск поможет решить эту задачу, выбрав из всех элементов только те, которые удовлетворяют поисковому запросу.
3. Выставление связи с другими слоями разрабатываемой системы. В ней будут присутствовать как минимум слой данных и слой работы с видео. Данные будут храниться в распределенной базе данных, но в DAM пользователь должен видеть необходимую ему информацию.
4. Работа со сторонними сервисами. Онлайн курсы обычно находятся на привычных для студентов порталах, таких как Coursera или Stepik. Видео можно хранить на таких сервисах как youtube.

# **Глава 2. Проектирование**

В качестве проекта рассмотрим следующую mind map:Указанные компоненты необходимо описать для понимания формата файлов и сервисов для включения их в систему.

## **Компоненты**

Автоматизированные рабочие места (АРМ) сопоставляются каждой роли, перечисленой во введении. В каждом АРМ создаются различные файлы.

1. Автор:
   1. Презентации. Для них накладывается ограничение: текст должен быть копируемым. Формат: pptx, ppt или pdf. Некоторые презентации формируются с помощью пакета LaTeX Beamer. Для них добавляется ещё tex файл. Одному модулю соответствует один документ.
   2. Соглашения подписываются автором, сканируются и присылаются в центр.
   3. Сценарии являются Microsoft Word документом. В нём указывается опорный текст и ключевые слова для переключения слайдов.
   4. Правки: после создания презентаций и видео в центре, автор получает их для просмотра и поиска ошибок. Они оформляются в виде Microsoft Excel или Word файла.
   5. Ссылки или дополнительные материалы помогают слушателям подробнее изучить преподаваемый в курсе материал. Автор может как сослаться на общедоступные источники в сети, так и создать собственный конспект в любом формате.
   6. Структура определяет количество модулей, темы, задания, дополнительные материалы. Связывает все остальные файлы. Имеет формат Microsoft Word документа.
2. Директор принимает различные соглашения, регулирующие работу центра. Они сканируются и хранятся в формате изображения.
3. Правки методиста оформляется по тому же принципу, что и правки автора.
4. Дизайнер:
   1. Промежуточные презентации являются первыми версиями презентаций, созданных дизайнером. Они не отправляются автору, а проверяются внутри центра методистом для внесения правок.
   2. Итоговые презентации создаются в результате работы над промежуточными версиями. Презентация становится итоговой после того, как у автора не остаётся замечаний к презентации.
5. Оператором и всей съёмочной группой проводятся съёмки видео.
6. Монтажёр объединяет итоговую презентацию со снятым видео. Для этого он использует данные структуры и сценария курса. Аналогично с презентациями, монтаж может быть промежуточным и итоговым.

Для разработки, хранения и публикации материалов курса используются различные сервисы и платформы. Они указаны в схеме. Для всех компонент указаны хосты и сети. Отдельно выделены компоненты хранения данных. Так как в центре создаются большие объёмы видео, для них необходимо место хранения. В качестве такового используются облачные сервисы Яндекс.Диск и Google Drive. Некоторые видео размещаются на YouTube. Однако основная часть материалов располагается на сервере СПбГУ.

## **Процесс создания курса**

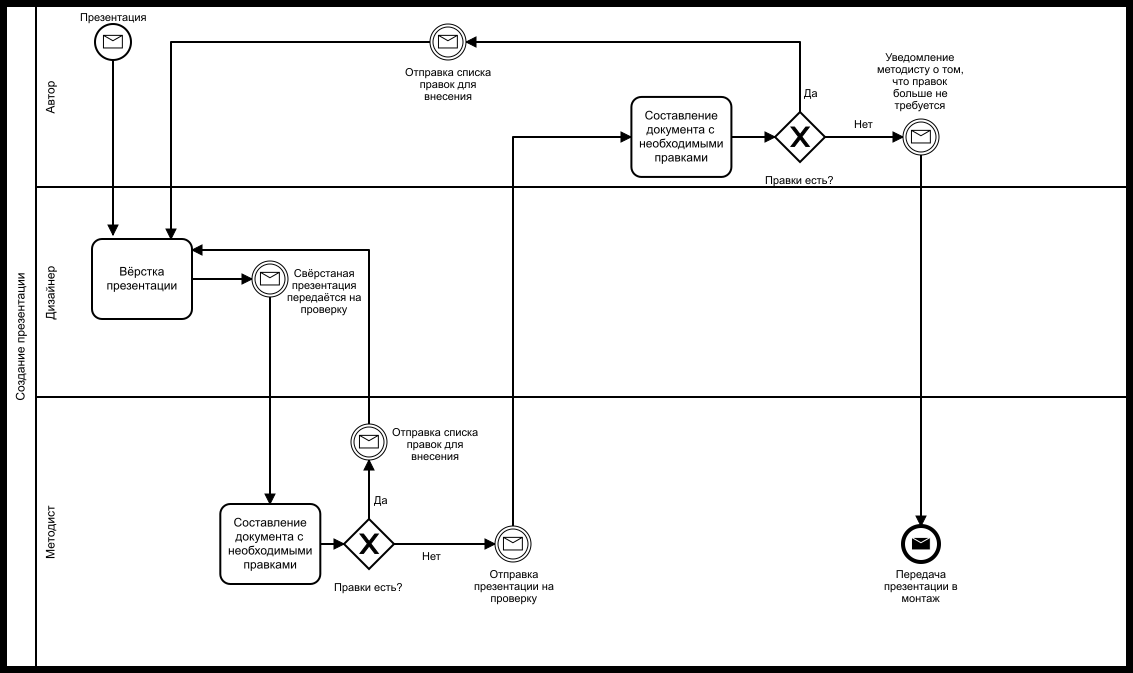
Процесс создания курса достаточно большой. Все действия имеют последовательность.

1. Директор выдает распоряжение о создании курса
2. Курс назначается методисту
3. Автор присылает структуру, сценарий и презентацию для каждого модуля
4. Создание презентаций (см. 3.3.)
5. Директор назначает дату съёмок
6. Режиссёры проводят видеосъемку автора
7. Создание видео (см. 3.4.)
8. Автор присылает дополнительные материалы
9. Автор или методист загружает курс на платформу
10. Автор подписывает соглашение и присылается в отсканированном виде методисту
11. Автор проверяет курс
12. Автор следит за форумом или обсуждениями курса
13. Методист настраивает экзамен
14. Студенты сдают экзамен
15. Выставление оценок и формирование сертификатов на их основе

Некоторые действия являются процессами, рассмотрение которых требует отдельных параграфов.

## **Процесс создания презентации**

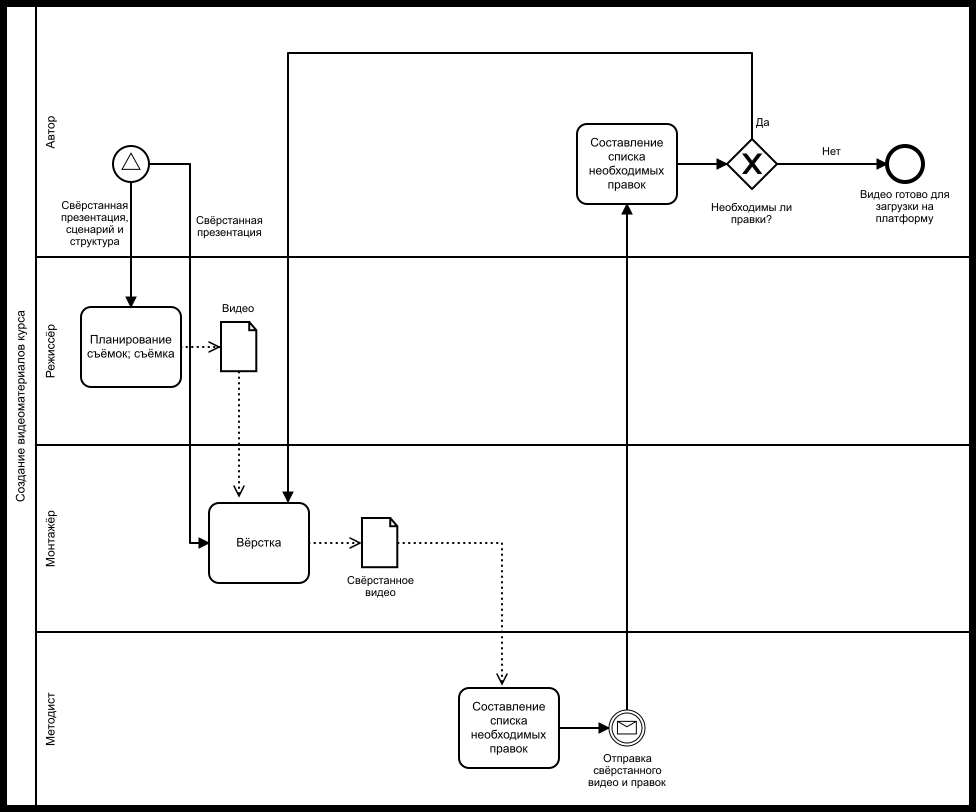
Моделирование процессов представлено в виде BPMN диаграмм. На ней отмечены роли, участвующие в разработке, их взаимодействия и связи. Также видна последовательность действий.



Как видно из диаграммы, в процессе работы презентации необходимо выделить 3 роли: дизайнер, методист и автор. Презентация проходит через несколько состояний:

* Прислана автором
* Промежуточная вёрстка
* Ждёт внесения изменений
* Итоговая презентация

## **Процесс создания видео**

Аналогично с процессом вёрстки презентации, процесс создания видео оформлен в виде BPMN диаграммы. 

В этом процессе выделяется 4 роли:

* Автор
* Режиссёр
* Монтажер
* Методист

# 

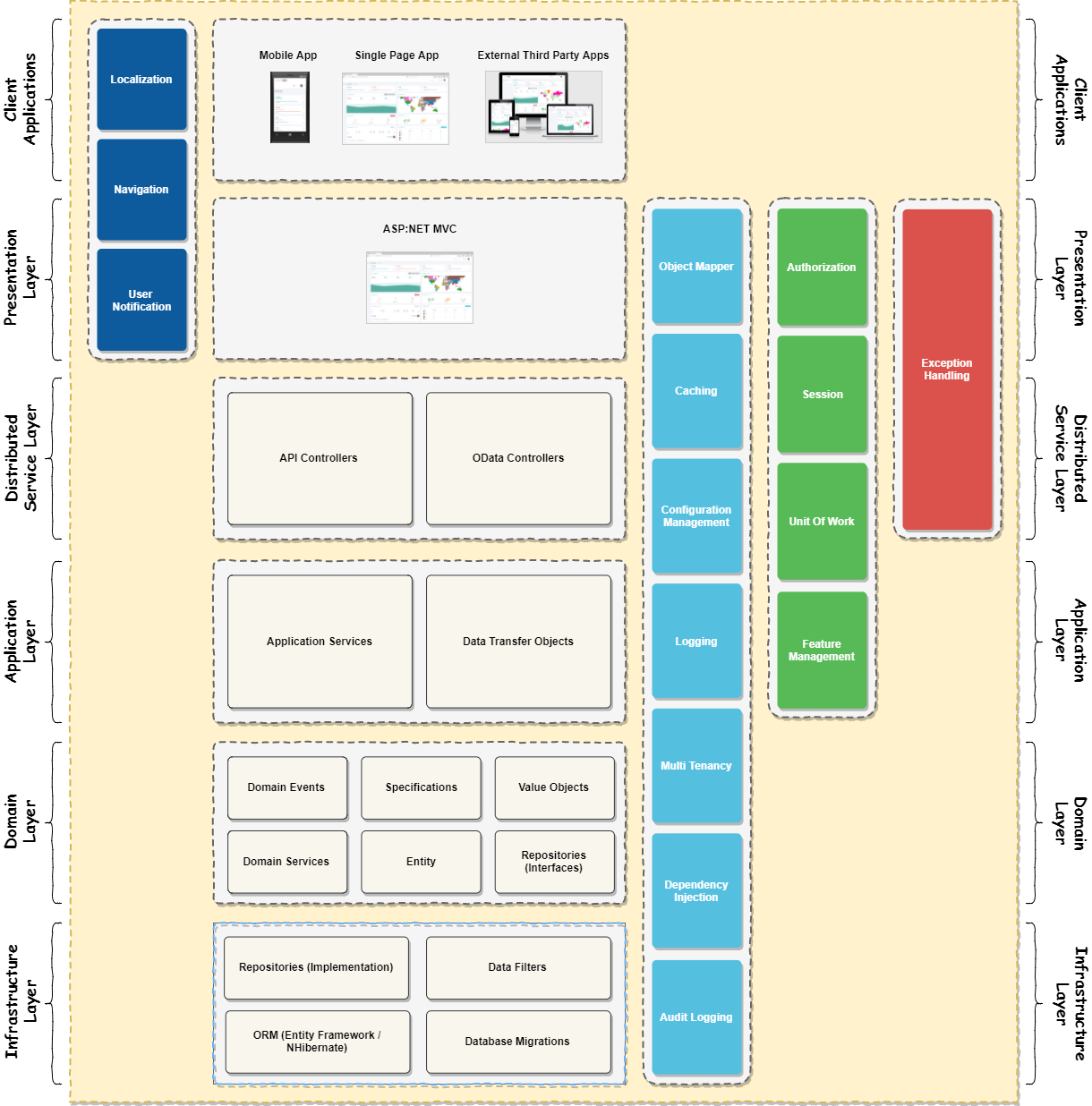
# **Глава 3. Разработка прототипа**

Прежде чем приступать к разработке прототипа, необходимо определиться с выбором технологий.

**3.1. Выбор технологий**

В качестве архитектурного фреймворка был взят ASP.NET Boilerplate (ABP). Рассмотрим его особенности:

* Многослойная архитектура. Так как на основе прототипа будет создаваться приложение, то оно должно быть масштабируемым. Выделение слоёв упрощает эту задачу. Например, в прототипе используется небольшая библиотека в качестве системы управления базами данных (СУБД), но в дальнейшем она будет заменена на полноценную СУБД.



* Мультиарендность. Наличие такого инструментария поможет преобразовать разрабатываемое ПО в SaaS.
* Модульный дизайн. Разбиение проекта на слои помогает в разработке тем, что любой компонент можно заменить на другой. Таким образом можно менять используемые сервисы и компоненты. Формат курса периодически меняется, из-за чего в систему придётся вносить изменения.

В качестве Web API взят ASP.NET; для разработки front-end Angular2. Такой выбор был сделан по причине того, что эти технологии знакомы разработчику.

## **3.2. Создание проекта**

Рассмотрим алгоритм создания проекта, где в роли Web API выступает ASP.NET, а в роли front-end (Application Layer) – Angular2:

1. Создать проект ASP.NET MVC + Web API.
2. Сохранить пустое Angular приложение: <https://github.com/angular/quickstart>. Из корневой папки перенести файл package.json в корневую директорию создаваемого проекта. Из папки src перенести файлы main.ts, systemjs-angular-loader.js, systemjs.config.extras.js, systemjs.config.js, tsconfig.json и папку app в корневую директорию создаваемого проекта.
3. Изменить view, которое будет представлять angular приложение так, чтобы оно было копией index.html из репозитория упомянутого выше. При необходимости изменить путь до файлов (href).
4. Установить Microsoft.TypeScript.MSBuild пакет. Его наличие понадобится в будущем. Необходима настройка файла решения в соответствии со ссылкой: <https://stackoverflow.com/questions/40441320/visual-studio-team-services-build-typescript-error>

После успешного прохождения всех шагов, приложение готово для размещения его в Visual Studio Team Services для его дальнейшего build & deploy.

## **3.3. Создание build**

Для успешной сборки созданного проекта в visual studio team services потребуется создать определение билда, которое предназначено для сборки ASP.NET Standart приложений. Шаги, которые должны присутствовать для сборки:

1. NuGet Tool Installer
2. NuGet restore
3. Npm install. Значение “Working folder with package.json” должно содержать путь до директории с файлом package.json, скопированного из angular quick start
4. Visual Studio Build (созданный проект по инструкциям, данным в прошлом параграфе, собирается со стандартными значениями параметров данного шага)
5. Copy Files. Необходимо настроить три параметра:
   1. source folder -“$(Build.SourcesDirectory)”
   2. Contents - “\*\*\node\_modules\\*\*”
   3. Target folder - “$(Build.ArtifactStagingDirectory)\dependecies”
6. Publish build artefacts

Такая конфигурация сборки совершит следующие действия: установку зависимостей проекта, их сборку и публикацию результата компиляции. Ключевым элементом является использование Microsoft.TypeScript.MSBuild, так как он отвечает за компиляцию TypeScript кода, который лежит в основе выбранного front-end инструментария.

## **Создание Release**

Далее будет описан процесс публикации сборки на Azure по подписке DreamSpark, доступной студентам. Первым шагом необходимо создание приложения на портале Azure (Azure Web App). Используя данные о созданном приложении, можно создать определение для Release. Шаги:

1. Azure App Service Deploy
2. Второй шаг аналогичен первому, но значение Package or folder необходимо заменить на $(System.DefaultWorkingDirectory)/<название сборки>/drop/dependecies/DERC/DERC

Таким образом на Azure отправляются файлы не только сгенерированные в результате msbuild, но и node\_modules.

**3.5. Настройка СУБД**

В ABP в качестве Object-Relational Mapping (ORM) используется Entity Framework (EF). Это необходимо для доступа к базе данных (БД). Так как на данном этапе решается задача разработки прототипа, а не полноценной системы, то вместо полноценной СУБД используется лёгкая. Базы данных делятся на реляционные и нереляционные. Второй тип больше подходит для разработки прототипа. Под такое описание попадает .NET библиотека LiteDB. Чтобы подключить её к проекту, необходимо установить соответствующий NuGet пакет.

Все настройки, которые будут описаны в этом параграфе, необходимо производить в отдельном слое. В нём необходимо создать модуль – класс, наследующий AbpModule.

Обращение к базе данных происходит следующим образом:

using (var db = new LiteDatabase(path)

{

var collection = db.GetCollection<Type>(dbName);

collection.Insert(entity);

return entity;

}

Данный код обращается к базе данных по пути path (если бд не существует, то она создаётся по указанному адресу). В этой базе данных создаётся коллекция для типа Type и ей присваивается название dbName. С помощью объекта collecton в базу данных можно вносить и извлекать данные. В качестве типа данных используются Entity, как этого требует паттерн Repository

Чтобы использовать эту базу данных в других слоях, необходимо задать правило генерации репозитория. В ABP они наследуются от типа IRepository<>. Если бы использовался встроенный в шаблон ORM, то репозитории бы генерировались автоматически, однако в выбранной ситуации необходимо провести настройку.

В первую очередь необходимо создать класс, от которого будут наследоваться все репозитории:

public class FakeRepositoryBase<TEntity, TPrimaryKey> : AbpRepositoryBase<TEntity, TPrimaryKey> where TEntity : class, IEntity<TPrimaryKey>

в котором нужно определить все необходимые методы, которые он наследует. Таким образом будет описана база для всех репозиториев типа IRepositry<TEntity, TPrimaryKey>, которые будут использоваться в дальнейшем. Чтобы была возможность использовать репозитории типа IRepository<TEntity>, необходимо унаследовать от созданной базы ещё один класс следующим образом:

public class FakeRepositoryBase<TEntity> : FakeRepositoryBase<TEntity, int>, IRepository<TEntity> where TEntity : class, IEntity<int>

{

public FakeRepositoryBase() : base()

{

}

}

В этом классе не нужно производить никаких настроек, так как они уже реализованы в базе.

Следующим шагом необходимо зарегистрировать новые зависимости. С этой задачей справляется следующий код:

public class RepositoriesInstaller : IWindsorInstaller

{

public void Install(IWindsorContainer container, IConfigurationStore store)

{

container

.Register(Component.For(typeof(IRepository<>))

.ImplementedBy(typeof(FakeRepositoryBase<>))

.LifestyleTransient()

);

}

}

Регистрация происходит в методе Initialize() модуля слоя данных:

IocManager.IocContainer.Register(Component.For(typeof(IRepository<>)).ImplementedBy(typeof(FakeRepositoryBase<>)));

В результате выполнения всех действий этого параграфа во всех слоях платформы появляется возможность использовать репозитории, с помощью которых можно обращаться к базе данных. Для этого необходимо создать объект типа IRepository<EntityName>, где EntityName является типом Entity. Они объявляются в слое Domain.

**3.6. Настройка слоя Application**

Слой Application необходим для отображения логики слоя Domain пользователю.

# Тесты