# Аннотация

Дипломный проект посвящен разработке и реализации программного продукта «Платформа для обучения программированию на Java».

В первом разделе проведен сравнительный анализ аналогичных программных продуктов предназначенных для обучения, описано назначение и сформулированы требования к разрабатываемому программному продукту.

# СОдержание

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ**

ПО – программное обеспечение

АСУ – автоматизированная система управление

УМК – учебно-методический комплекс

ЭУМК – электронный учебно-методический комплекс

RAD – Rapid Application Development (модель быстрой разработки приложений)

4GL – Fourth Generation Language (язык программирования четвертого поколения)

CASE-средства – Computer Aided Software Engineering (компьютерная поддержка проектирования ПО)

SADT – Structured Analysis And Design Technique (методология структурного анализа и проектирования)

DFD – Data Flow Diagrams (методология структурного анализа потоков данных)

JVM – Java Virtual Machine (виртуальная машина Java)

HTML – HyperText Markup Language (язык гипертекстовой разметки)

CSS – Cascading Style Sheets (каскадные таблицы стилей)

JPA – Java Persistence API

API – application programming interface (программный интерфейс приложения)

ORM – Object-Relational Mapping (объектно-реляционное отображение)

DAO – Data Access Object (объект доступа к данным)

CRUD-операции – create, read, update, delete (создать, прочесть, обновить, удалить)

**ВВЕДЕНИЕ**

В современном мире, по причине быстрого развития информационных технологий, происходит их повсеместное внедрение во все сферы человеческой жизни. Также это происходит и в образовательном процессе. Тенденция развития информационных технологий и повсеместного внедрения Интернета приводит к тому, что образование перейдет полностью в Интернет на форму удаленного обучения. В настоящее время можно найти множество различных ресурсов для удаленного получения образования. Множество сертификатов мирового уровня можно пройдя тестирование находясь в любой точке земного шала.

В связи с этой тенденцией в данном дипломном проекте разрабатывается платформа для обучения программированию на Java. Целью проекта является повышение доступности и качества образовательной деятельности учебного заведения.

Перечень средств разработки:

– Средства проектирования приложения: IBM Rational Rose Enterprise Edition, Microsoft Visio 2010;

– Язык программирования: Java8;

– Фреймворки: Spring Framework, Spring Boot

– Средства создания пользовательского интерфейса: HTML, CSS;

– Средства получения исполняемого кода: IntelliJ IDEA 2017.3.5 x64;

– Средства отладчики исполняемого кода: IntelliJ IDEA 2017.3.5 x64;

– Средства тестирования программ: IntelliJ IDEA 2017.3.5 x64;

– Cредства управления базами данных: MySQL;

– Средства запуска готового проекта: JVM.

# 1 Описвние предметной области, Целей и назначения проекта

## 1.1 Цели разработки

Платформа предназначена для повышения доступности и качества образовательной деятельности в рамках учебного заведения, а также призвана облегчить работу преподавательской деятельности.

## 1.2 Основание для разработки

Разработка системы осуществляется на основании темы к дипломному проекту в соответствии с документацией деканата факультета «Электросвязи» Белорусской государственной академии связи и регламентируется Уставом Белорусской государственной академии связи.

## 1.3 Назначение

Назначение разрабатываемого программного продукта заключается в передаче студентам теоретических знаний концепции объектно-ориентированного программирования, практических навыков по разработке объектно-ориентированных программ на одном из популярных языков программирования – Java.

## 1.4 Описание предметной области

Опишем понятия, которые входят в предметную область выбранной темы.

**Программное обеспечение (ПО)** представляет собой программу, либо совокупность программ, предназначенных для управления компьютером.

Программное обеспечение в зависимости от назначения разделяется на: системное ПО, прикладное ПО и инструментальное ПО.

Системное программное обеспечения включает в себя программы, обеспечивающие функционирование компьютера и компьютерных сетей. Данное ПО позволяет пользователю осуществлять управление компьютером и компьютерной сетью, а также обеспечивает работоспособность других программ. В качестве примеров системного ПО можно привести операционные системы, архиваторы, файловые менеджеры, утилиты и другие.

Программы, предназначенные для решения задач определенной предметной области, относятся к прикладному программному обеспечению.

Прикладное ПО принято разделять на:

– программы общего назначения (необходимы для работы любого пользователя);

– программы специального назначения (требуются для использования профессионалами в различных сфера деятельности);

– компьютерные игры.

Примерами прикладных программ являются текстовые редакторы, графические редакторы и другие.

Инструментальное программное обеспечение предназначения для проектирования, разработки и сопровождения программ.

Различают такие виды инструментального ПО как: компиляторы, интерпретаторы, отладчики, SDK, интегрированные среды разработки и другие.

**Автоматизированные системы управления (АСУ)** – это комплекс программных и аппаратных средств, предназначенный для управления различными процессами в рамках технологического процесса, предприятия, производства. АСУ применяются в различных отраслях промышленности, энергетике, транспорте и т. п.

Наиболее важной задачей автоматизированных систем управления является повышение эффективности управления объектом на основе роста производительности труда и совершенствования методов планирования процесса управления.

Различают автоматизированные системы управления объектами (технологическими процессами – АСУТП, предприятием – АСУП, отраслью – ОАСУ) и функциональные автоматизированные системы, например, проектирование плановых расчётов, материально-технического снабжения и т.д.

**Java** является универсальным языком программирования, который основан на концепции объектно-ориентированного программирования. Он предназначен для того, чтобы позволить разработчикам приложений «писать один раз, выполнять где угодно» (WORA), что означает, что скомпилированный код Java может работать на всех платформах, поддерживающих Java, без необходимости перекомпиляции.

**Учебная программа** (Учебный план) учреждения образования – созданный в рамках системы обучения документ, определяющий содержание и количество знаний, умений и навыков, предназначенных к обязательному усвоению по той или иной учебной дисциплине, распределение их по темам, разделам и периодам обучения.

**Учебно-методический комплекс** (УМК) дисциплины – стандартное название для совокупности учебно-методической документации, средств обучения и контроля, разрабатываемых в учреждениях образования для каждой дисциплины. УМК должен включать полную информацию, достаточную для прохождения дисциплины. УМК предназначены для обеспечения открытости образовательного процесса и должны быть доступны любому желающему.

**Электронный учебно-методический комплекс** (ЭУМК) дисциплины – продукт, имеющий электронный формат, который используется в учебном процессе учебного заведения в рамках дисциплины.

**Контроль знаний** – оценка знаний учащихся по истечении конкретного периода учебного года (четверти, полугодия), позволяющая определить качество и объем усвоенных знаний.

**Журнал успеваемости** – документ, позволяющий участникам учебного процесса вести учет о текущих и итоговых оценках.

В качестве предметной области выбрано учреждение образования «Белорусская государственная академия связи», которое занимается обучением будущих специалистов в области телекоммуникаций, а точнее кафедра программного обеспечения сетей телекоммуникаций (ПОСТ) [1].

Кафедра является профилирующей и осуществляет подготовку специалистов на уровне высшего образования по направлению специальности 1-45 01 02-03 «Инфокоммуникационные системы (сопровождение программного обеспечения)», (квалификация специалиста – инженер по инфокоммуникационным системам) [1].

Кафедра осуществляет подготовку специалистов на уровне среднего специального образования по специальности 2-45 01 33 «Сети телекоммуникаций», специализации 2-45 01 33 02 «Программное обеспечение сетей телекоммуникаций» (квалификация специалиста – техник по телекоммуникациям) [1].

На уровне высшего образования преподается дисциплина «Конструирование программ и языки программирования», цель которой является приобретение студентами теоретических знаний концепции объектно-ориентированного программирования, практических навыков по разработке объектно-ориентированных программ, а также умений объектно-ориентированных программ [2].

В рамках дисциплины предусматривается изучение языка Java SE, аспекты применения библиотек классов языка Java, включая файлы, коллекции, сетевые и многопоточные приложения, а также взаимодействие с XML [2].

# 2 Выбор жизненного цикла проекта, анализ предметной области, выявление требований к системе

## 2.1 Выбор модели жизненного цикла проекта

Существуют различные схемы классификации проектов, связанных с разработкой программных средств и систем.

В **ГОСТ Р ИСО/МЭК ТО 12182–2002 – Информационная технология. Классификация программных средств** [3] – приведена схема классификации ПС по 16 видам, каждый из которых подразделяется на классы. Данная классификация имеет общий характер и в целом не может использоваться для обоснования выбора модели ЖЦ ПС и систем [4].

Институтом качества программного обеспечения SQI (Software Quality Institute, США) специально для выбора модели ЖЦ предложена схема классификации проектов по разработке ПС и систем [5]. Основу данной классификации составляют четыре категории критериев. По каждому критерию проекты подразделяются на два альтернативных класса [4].

Критерии классификации проектов, предложенные Институтом SQI, объединены в следующие категории.

*1. Характеристики требований к проекту*

Критерии данной категории классифицируют проекты в зависимости от свойств требований пользователя (заказчика) к разрабатываемой системе или программному средству. Например, известны ли требования к началу проекта, сложны ли они, будут ли они изменяться.

*2. Характеристики команды разработчиков*

Чтобы иметь возможность пользоваться критериями данной категории классификации проектов, состав команды разработчиков необходимо сформировать до выбора модели ЖЦ. Характеристики команды разработчиков играют важную роль при выборе модели ЖЦ, поскольку разработчики несут основную ответственность за успешную реализацию проекта. В первую очередь следует учитывать квалификацию разработчиков, их знакомство с предметной областью, инструментальными средствами разработки и т.п.

*3. Характеристики пользователей (заказчиков)*

Чтобы иметь возможность пользоваться критериями данной категории классификации проектов, до выбора модели ЖЦ необходимо определить возможную степень участия пользователей (заказчиков) в процессе разработки и их взаимосвязь с командой разработчиков на протяжении проекта. Это важно, поскольку отдельные модели ЖЦ требуют усиленного участия пользователей в процессе разработки. Выбор таких моделей при отсутствии реальной возможности пользователей участвовать в проекте приведет к существенному снижению качества результатов разработки.

*4. Характеристики типов проектов и рисков*

В некоторых моделях в достаточно высокой степени предусмотрено управление рисками. В других моделях управление рисками вообще не предусматривается. Поэтому при выборе модели ЖЦ следует учитывать реальные риски проекта, критичность и сложность продуктов разработки.

Для выбора подходящей к условиям конкретного проекта модели ЖЦ ПС и систем Институтом качества программного обеспечения SQI рекомендуется использовать специальную процедуру [5]. Данная процедура базируется на применении четырех таблиц вопросов. Примеры вопросов приведены в таблицах A.1-A.4 приложения А.

Для моего проекта наиболее важными вопросами из таблиц А.1-А.4 являются:1, 2, 3, 6, 7 – вопросы из таблицы А.1; все вопросы из таблицы А.2; 3, 4 – вопросы из таблицы А.3; 3, 5, 6, 7, 8, 10 – вопросы из таблицы А.4.

Таблица 2.1 – Итоги выбора жизненного цикла

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Количество отмеченных вопросов | Каскадная | Vобразная | RAD | Инкре-  ментная | Быстрого  прототипи-  рования | Эволюци-  онная |
| Важные | 14 | 13 | 14 | 8 | 7 | 5 |
| Не важные | 2 | 2 | 5 | 1 | 6 | 4 |
| Итого | 16 | 15 | 19 | 9 | 13 | 9 |

Исходя из итогов таблицы 2.1 лидирует модель жизненного цикла RAD по количеству важных вопросов и суммарному количеству вопросов. Так как к процессу разработки нет ограничений, выбираем **модель жизненного цикла RAD**.

## 2.2 Моделирование и анализ предметной области

В качестве модели жизненного цикла я выбрал модель RAD. Как правило, RAD-модели используются в составе другой модели для ускорения цикла разработки прототипа (версии) системы или программного средства. Но так как проект не является сложным, буду применять RAD-модель как независимую модель разработки.

Основу RAD-модели составляет использование мощных инструментальных средств разработки. Такими средствами являются языки четвертого поколения 4GL (Fourth Generation Language – язык программирования четвертого поколения) и CASE-средства (Computer Aided Software Engineering – компьютерная поддержка проектирования ПО), благодаря наличию в них сред визуальной разработки и кодогенераторов.

Использование инструментальных средств позволяет задействовать пользователя, а следовательно, дать оценку продукту на всех этапах его разработки.

Характерной чертой RAD-модели является короткое время перехода от анализа требований до создания полной системы или программного средства. Разработка прототипа, как правило, ограничивается четко определенным периодом времени.

Существует 3 основные RAD-модели:

– *Базовая RAD-модель* применяется совместно с другой моделью разработки;

– *RAD-модель, основанная на моделировании предметной области* – используются

различные виды моделирований предметной области, например, функциональное моделирование, моделирование данных, моделирование процесса (поведения). С этой целью широко используются CASE-средства;

– *RAD-модель параллельной разработки приложений* – данный вариант модели поддерживает параллельную разработку программных компонентов, реализующих базовые функции программного средства различными группами разработчиков, с дальнейшей интеграцией разработанных компонентов в единую систему или программное средство.

Исходя из описания, базовая RAD-модель и RAD-модель параллельной разработки приложений не подходят, так как базовая модель учитывает, что она будет использоваться с другой моделью разработки, чего я не предусматриваю. RAD-модель параллельной разработки приложений не подходят, так как в ней должны участвовать группа разработчиков, что в данном случае невозможно. Следовательно, я выбрал **RAD-модель, основанную на моделировании предметной области**.

В данной модели выделяется пять этапов (рисунок 2.1).

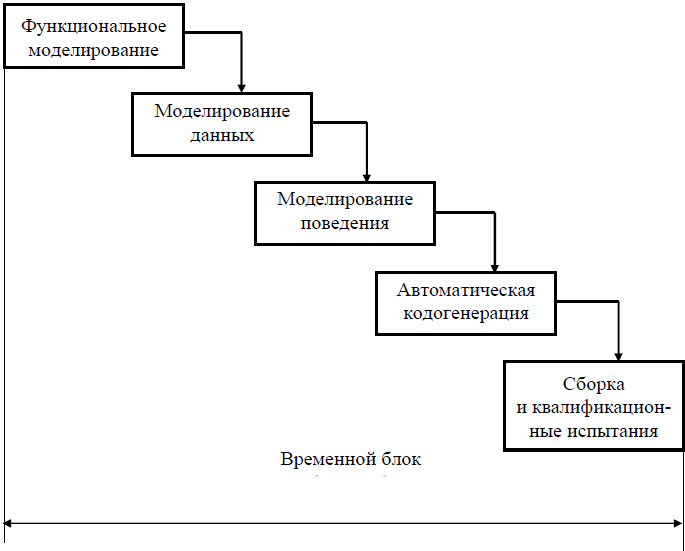


Рисунок 2.1 – Вариант модели быстрой разработки приложений, основанной на моделировании предметной области

*На этапе функционального моделирования* определяются и анализируются функции и информационные потоки предметной области, в которой используются методологии функционального моделирования IDEF0 и DFD.

*На этапе моделирования данных на базе информационных потоков*, определенных на предыдущем этапе, разрабатывается информационная модель предметной области в которой используется методология информационного моделирования IDEF1X.

*На этапе моделирования поведения* выполняется динамическое (поведенческое) моделирование предметной области. Одной из известных методологий динамического моделирования является методология IDEF3.

*На этапе автоматической кодогенерации* на основе информационной, функциональной и поведенческой моделей выполняется генерация текстов программных компонентов. При этом используются языки программирования четвертого поколения (Fourth Generation Language – 4GL) и CASE-средства. Широко применяются повторно используемые программные компоненты.

*На этапе сборки и квалификационных испытаний* выполняется сборка и испытания результирующей системы, подсистемы или программного средства.

**2.2.1** Этап функционального моделирования.

Основой для методологии функционального моделирования IDEF0 является методология структурного анализа и проектирования SADT. Основным назначением методологии SADT является моделирование предметной области с целью определения требований к разрабатываемой системе или программному средству и с целью их проектирования.

В моделировании SADT определены два направления: *функциональные модели* (IDEF0) выделяют события в системе, *модели данных* (DFD) выделяют объекты (данные) системы, связывающие функции между собой и с их окружением.

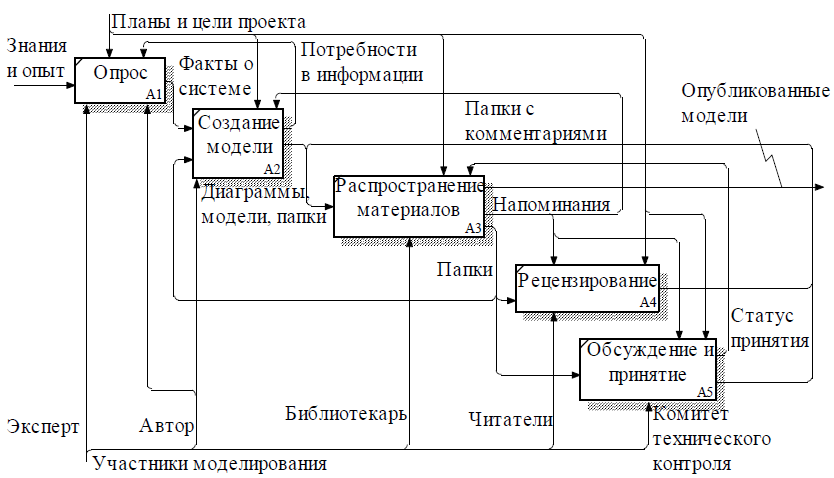


Рисунок 2.2 – Процесс моделирования в IDEF0

**На первом этапе** разработки IDEF0-модели формулируются вопросы к ней, формируется цель модели, определяются претенденты на точку зрения, выбирается точка зрения.

*Перечень вопросов*:

– Какие этапы обучения необходимо пройти студенту?

– Какие сотрудники учувствуют в процессе обучения и аттестации студентов?

– Какие виды работ должен осуществлять преподаватель во время обучения студентов и их аттестации?

– Какая информация должна быть входной для обучения студентов?

– Как влияют результаты отдельных этапов на итоги обучения студента?

– Что необходимо для завершения обучения студентом?

*Цель модели:* определить основные этапы прохождения обучения студентами, влияние этапов друг на друга и на результаты завершения обучения с целью обучения студентов концепциям объектно-ориентированного программирования на Java.

*Претенденты на точку зрения:* преподаватель, студент.

С учетом цели модели предпочтение следует отдать точке зрения преподавателя, так как она наиболее полно охватывает все этапы обучения и только с этой точки зрения можно показать взаимосвязи между отдельными этапами и обязанности участников обучения.

**Вторым этапом** моделирования является создание модели. На данном этапе полученные на предыдущем этапе факты о системе автор представляет в виде одной или нескольких IDEF0-диаграмм. Процесс создания модели осуществляется с помощью метода декомпозиции ограниченного субъекта. При его использовании автор модели вначале анализирует объекты (информацию, данные, механизмы и т.п.), входящие в систему, а затем использует полученные знания для анализа функций системы (см. п. 5.2.5). На основе этого анализа создается диаграмма, в которой объединяются сходные объекты и функции.

IDEF0 диаграммы представлены в приложении Б.

Третий, четвертый и пятый этапы моделирования можно пропустить, так как эти этапы не являются информативными и не несут важных сведений. Целю моделирования было получение полного представления о процессах обучения в учебном заведении, что было достигнуто.

Методология структурного анализа потоков данных DFD (Data Flow Diagrams) основана на методах, ориентированных на потоки данных. Существуют различные графические нотации данной методологии. В данном проекте используется методология DFD в нотации Гейна–Сарсона.

DFD-методология выделяет функции (действия, события, работы) системы. Функции соединяются между собой с помощью потоков данных (объектов).

Аналогично IDEF0-методологии DFD-модель должна иметь единственные цель, точку зрения, субъект и точно определенные границы. Однако если в IDEF0 дуги имеют различные типы и определяют отношения между блоками, то в DFD дуги отражают реальное перемещение объектов от одной функции к другой.

*Цель модели:* определить основные этапы прохождения обучения студентами, влияние этапов друг на друга и на результаты завершения обучения с целью обучения студентов концепциям объектно-ориентированного программирования на Java.

*Претенденты на точку зрения:* преподаватель, студент.

С учетом цели модели предпочтение следует отдать точке зрения преподавателя, так как она наиболее полно охватывает все этапы обучения и только с этой точки зрения можно показать взаимосвязи между отдельными этапами и обязанности участников обучения.

IDEF0 диаграммы представлены на рисунках 2.3-2.6.

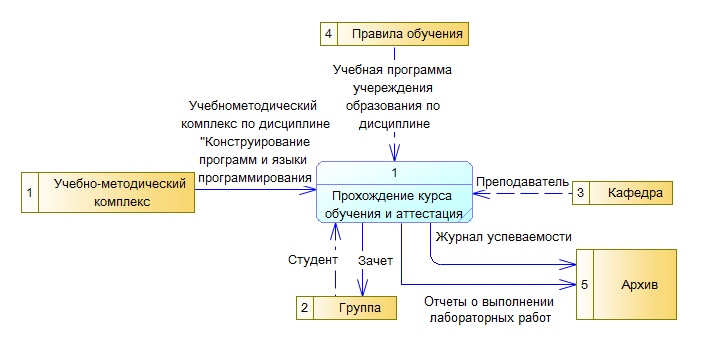


Рисунок 2.3 – Концептуальная схема

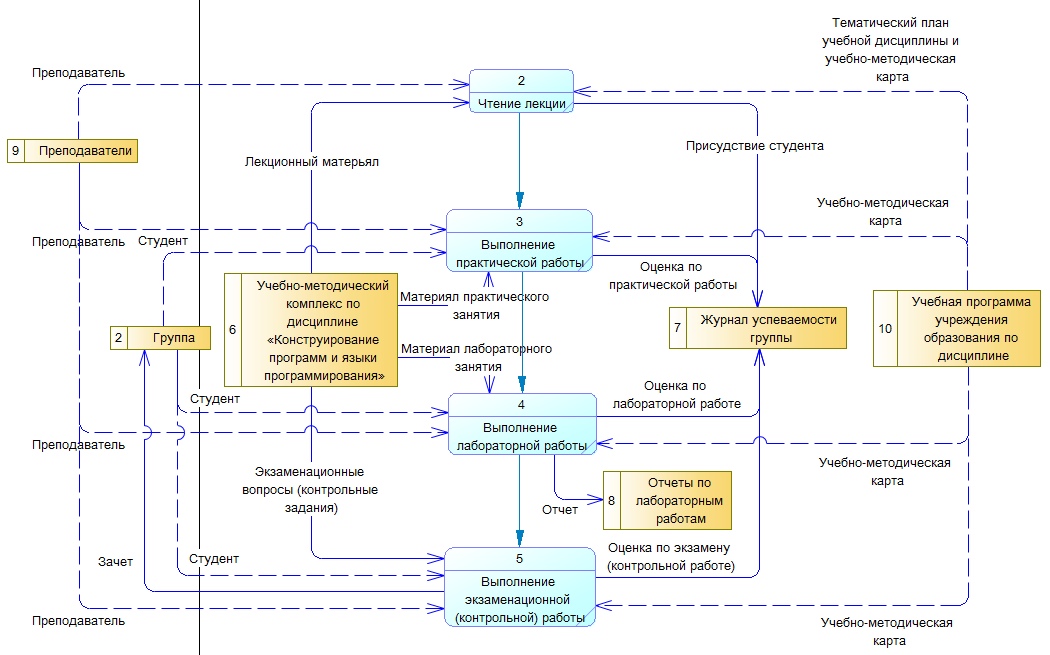


Рисунок 2.4 – Декомпозиция концептуальной схемы

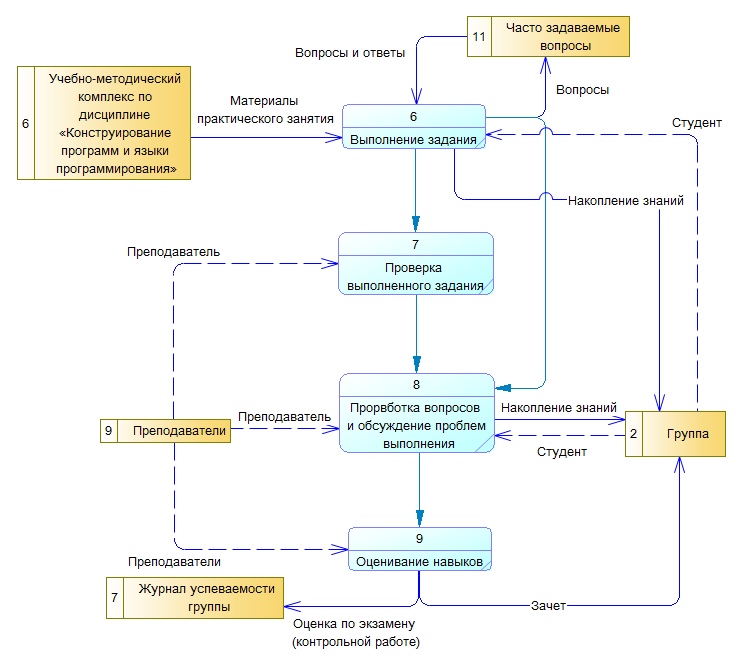


Рисунок 2.5 – Декомпозиция процесса выполнения практического задания

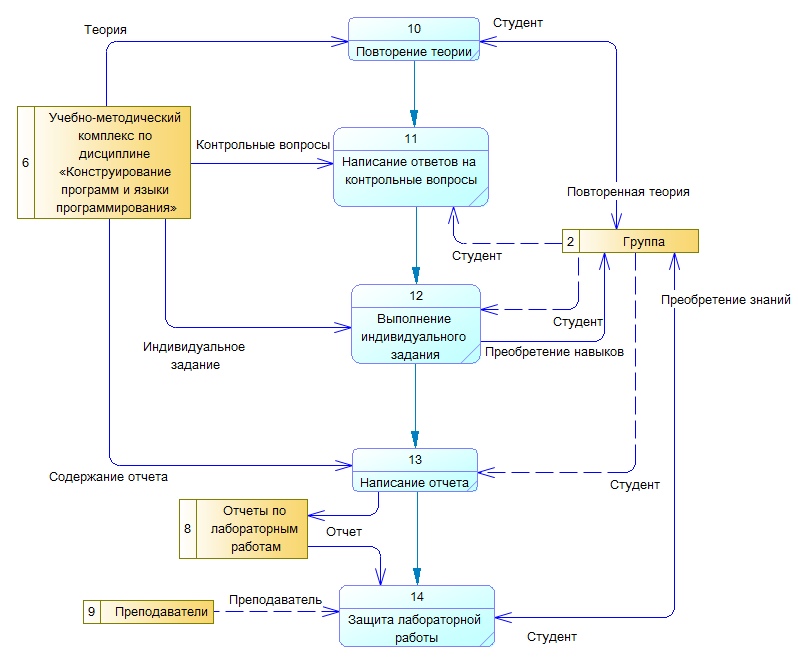


Рисунок 2.6 – Декомпозиция выполнения лабораторной работы

**2.2.2** Этап информационного моделирования

Методология информационного моделирования IDEF1X известна также под названиями методологии семантического моделирования данных [6] и методологии концептуального моделирования [7].

Под информационной моделью (моделью данных) подразумевается графическое и текстовое представление результатов анализа предметной области, которое идентифицирует данные, используемые в организации [7].

*Целью* этапа является выявление сущностей, составляющих предметную область, и отношений между ними. *Результатом* является информационная модель предметной области, содержащая сущности, их атрибуты и отражающая отношения между сущностями.

Результаты этапа информационного моделирования буду использоваться для проектирования базы данных разрабатываемой системы.

Общепринятым стандартом представления информационных моделей (моделей данных) в настоящее время является стандарт IDEF1X [7], разработанный на основе диаграмм «Сущность–Связь» Чена.

Информационная модель предметной области, разработанная на основе моделирования потоков данных DFD, представлена на рисунке 2.7.

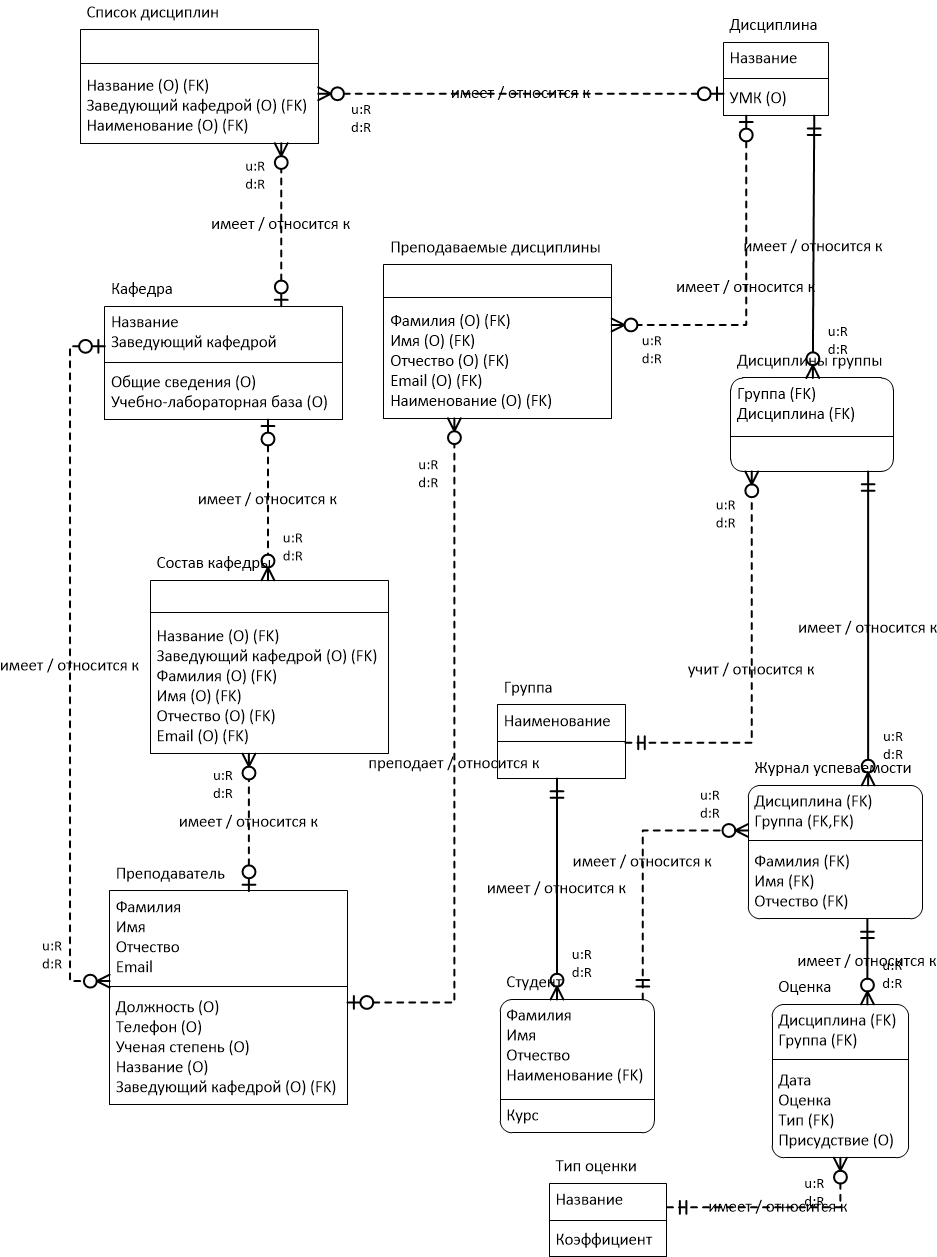


Рисунок 2.7 – Информационная модель предметной области IDEF1X

**2.2.3** Этап моделирования поведения.

Динамическое (поведенческое) моделирование предметной области я считаю можно пропустит, так как поведение предметной области функционально полно отражено на IDEF0 диаграммах в приложении Б.

## 2.3 Требования к системе

В этом подразделе классификация требований описана по К. Вигерсу [8]. Классификация разделяет требования к разрабатываемой системе на 3 уровня:

– *Бизнес-требования* определяют высокоуровневые цели организации или клиента;

– *Пользовательские требования* описывают цели/задачи пользователей системы, которые должны выполняться пользователями при помощи создаваемого ПО;

– *Функциональные требования* определяют функциональность ПО, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требования.

Классификация требований по К. Вигерсу представлена на рисунке 2.8.



Рисунок 2.8 – Классификация требований по К. Вигерсу

**2.3.1** Бизнес-требования.

Главной целю учебного заведения является передача и контроль знаний. В следствии чего выделяются следующие бизнес-требования:

– Система должна предоставлять теоретический материал в полном объеме в соответствии с УМК предоставляемым учебным заведением;

– Система должна учитывать учебный план учебного заведения и быть адаптивной к возникающим изменениям;

– В течении обучения система должна предоставлять помощь в контроле знаний участников обучения.

**2.3.2** Пользовательские требования.

Пользователями системы являются Преподаватели и Студенты.

Требования к системе для пользователя Студент:

– Система должна легко и удобно предоставлять доступ к изучаемому материалу;

– Система должна обеспечивать среду для выполнения тестовых и практических заданий;

– Система должна предоставлять возможность сдачи отчета о выполненной лабораторной работе и практических работ;

Требования к системе для пользователя Преподаватель:

– Система должна предоставлять инструменты для создания, модернизации, расширения и управлять базой данных материала обучения, тестовых и практических заданий, дополнительных материалов;

– Система должна хранить отчетность и выполненную работу студентов, и предоставлять доступ к ней;

– Система должна предоставлять инструменты для создания и управления учетными записями студентов и преподавателей, а также студенческими группами;

– Система должна позволять вести учет успеваемости студентов в течении их обучения.

**2.3.3** Функциональные требования

Из пользовательских требований вытекают следующие функциональные требования:

– В системе должен присутствовать конструктор курса обучения, с помощью которого преподаватель сможет создавать, модернизировать, расширять и управлять курсами и планом обучения;

– Система должна предоставлять графический интерфейс для получения доступа к материалам курса для обучения студентов;

– В системе должна присутствовать форма для отправки отчетности о выполнении лабораторных работ, практических заданий и курсовых проектов на проверку преподавателем с обратной связью;

– В системе должен присутствовать онлайн компилятор для выполнения практических заданий студентами;

– В системе должен присутствовать журнал ведения учета успеваемости студентов;

– После выполнения студентами тестовых заданий система должна проверить ответы студентов на правильность выполнения теста и выставить оценку;

Из этой классификации можно выделить еще системные требования:

– Web-приложение запускаемое любым контейнером приложений;

– Кроссплатформенное приложение.

# 3 ОПИСАНИЕ Средств разработки

## 3.1 Язык программирования и фреймворк

**Java** – сильно типизированный объектно-ориентированный кроссплатформенный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Исходный код, написанный на Java, транслируемый компилятором в байт-код, выполняется виртуальной машиной Java (JVM). JVM – программа, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор, но с тем отличием, что байт-код, в отличии от текста, обрабатывается значительно быстрее. Он предназначен для того, чтобы позволить разработчикам приложений «писать один раз, запускать где угодно» (WORA) [9], что означает, что скомпилированный код Java может работать на всех платформах, поддерживающих Java, без необходимости перекомпиляции [10].

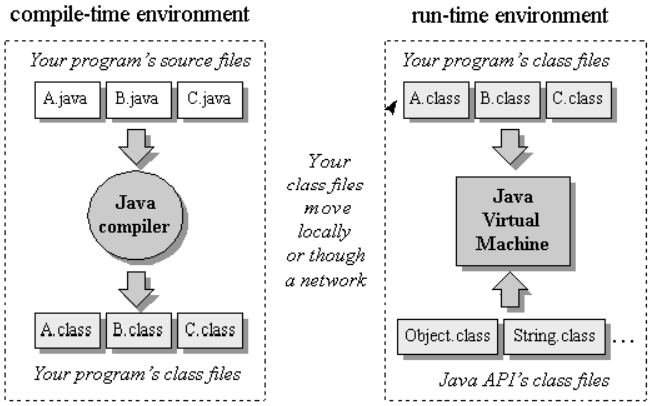


Рисунок 3.1 – Среда программирования Java

Также известно, что Java ориентирована на Internet. С помощью Java web-страницу можно наполнить не только обычным текстом, но и динамическими элементами.

Java-платформа обладает следующими преимуществами:

– Кроссплатформенность;

– Объектная ориентированность, создана эффективная объектная модель;

– Встроенная и прозрачная модель безопасности;

– Ориентация на Internet-задачи, сетевые распределенные приложения;

– Динамичность, легкость развития и добавления новых возможностей;

**Spring Framework** предоставляет комплексную модель программирования и конфигурации для современных корпоративных приложений на основе Java – на любой платформе развертывания [11].

Ключевым элементом Spring является инфраструктурная поддержка на уровне приложений: Spring фокусируется на «подключении» корпоративных приложений, чтобы группы могли сосредоточиться на бизнес-логике на уровне приложений без ненужных связей с конкретными средами развертывания.

Характеристика Spring Framework:

– Основные технологии: внедрение зависимостей, события, ресурсы, i18n, проверка, привязка данных, преобразование типов, SpEL, AOP.

– Тестирование: макет объектов, каркасные TestContext, Spring MVC Test, WebTestClient.

– Доступ к данным: транзакции, поддержка DAO, JDBC, ORM, Marshalling XML.

– Spring MVC и Spring WebFlux веб-фреймворки.

– Интеграция: удаленное взаимодействие, JMS, JCA, JMX, электронная почта, задачи, планирование, кэш.

**Spring Boot** позволяет легко создавать автономные производственные приложения на основе Spring, которые можно «просто запустить» [12].

Мы принимаем взвешенное мнение о платформе Spring и сторонних библиотеках, чтобы вы могли начать работу с минимальными усилиями. Большинству приложений Spring Boot требуется очень небольшая конфигурация Spring.

Характеристика Spring Boot:

– Создание автономных приложений Spring

– Внедрить Tomcat, Jetty или Undertow напрямую (не нужно развертывать файлы WAR)

– Предоставьте самоуверенные зависимости для начинающих, чтобы упростить конфигурацию сборки

– Автоматическая настройка Spring и сторонних библиотек, когда это возможно

– Обеспечить готовые к работе функции, такие как метрики, проверки работоспособности и внешняя конфигурация

– Абсолютно нет генерации кода и не требуется настройка XML

## 3.2 Средства получения, отладки исполняемого кода и средства тестирования программ

IntelliJ IDEA – интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains [13]. Обладает широким набором интегрированных инструментов для рефакторинга, которые позволяют программистам быстро реорганизовывать исходные тексты программ, и идеально подходит для создания коммерческих, мобильных и веб-приложений. IntelliJ IDEA включает поддержку всех последних технологий и фреймворков, работает с автоматизированными инструментами сборки и управления проектом, включа Maven, Gradle, Ant и другими. Также IntelliJ IDEA обладает встроенным отладчиком исполняемого кода и средствами тестирования программ.

## 3.3 Cредства управления базами данных

MySQL – свободная реляционная система управления базами данных. MySQL является решением для малых и средних приложений [14]. 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.

MySQL имеет API для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентный Паскаль и Tcl, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

## 3.4 Средства запуска готового проекта

В основе технологии Java лежит **виртуальная машина Java** (JVM) – абстрактный компьютер, на котором работают все программы Java [15] – основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java, Java API и файл классов Java работают вместе с языком для запуска программ Java.

Виртуальные машины Java обычно содержат Интерпретатор байт-кода, однако, для повышения производительности во многих машинах также применяется JIT-компиляция часто исполняемых фрагментов байт-кода в машинный код.

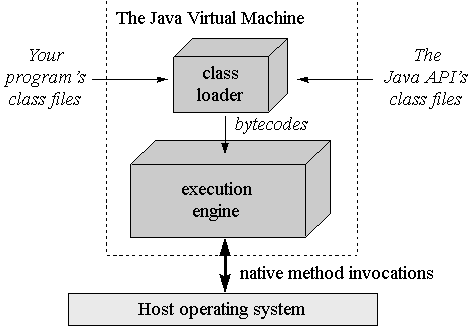


Рисунок 3.2 – Виртуальная машина Java, реализованная в программном обеспечении поверх операционной системы хоста

**Apache Tomcat** – контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF) [16]. Написан на языке Java.

Tomcat позволяет запускать веб-приложения, содержит ряд программ для самоконфигурирования.

Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

## 3.5 Средства проектирования приложения

Microsoft Visio – векторный графический редактор, редактор диаграмм и блок-схем для Windows. Основным назначением Visio является визуализация данных или, проще говоря, изготовление практически любых схем, иллюстраций и диаграмм, необходимых для оформления деловой документации.

Используя эту программу, вы можете отобразить ход выполнения проекта, подготовить план здания, составить информационную модель предметной области IDEF1X, составить функциональную модель IDEF0 и смоделировать информационные потоки DFD предметной области. Содержание иллюстраций зависит только от вашего воображения. При этом вы сможете гибко настраивать внешний вид элементов, что позволит не только использовать функциональную окраску частей иллюстрации, но и получить опрятный и привлекательный документ.

CASE-средство IBM Rational Rose со времени своего появления претерпело серьезную эволюцию, и в настоящее время представляет собой современный интегрированный инструментарий для проектирования архитектуры, анализа, моделирования и разработки программных систем. Именно в IBM Rational Rose язык UML стал базовой технологией визуализации и разработки программных систем, что определило популярность и стратегическую перспективность этого инструментария.

Наиболее характерные функциональные особенности этой программы заключаются в следующем:

– интеграция с MS Visual Studio 6, которая включает поддержку на уровне прямой и обратной генерации кодов и диаграмм Visual Basic и Visual С++ с использованием ATL (Microsoft Active Template Library), Web-Классов, DHTML и протоколов доступа к различным базам данных;

– непосредственная работа (инжиниринг и реинжиниринг) с исполняемыми модулями и библиотеками форматов EXE, DLL, TLB, OCX.

– поддержка технологий MTS (Microsoft Transaction Server) и ADO (ActiveX Data Objects) на уровне шаблонов и исходного кода, а также элементов технологии Microsoft - COM+ (DCOM);

– полная поддержка компонентов CORBA и J2EE, включая реализацию технологии компонентной разработки приложений CBD (Component-Based Development), языка определения интерфейса IDL (Interface Definition Language) и языка определения данных DDL (Data Definition Language);

– полная поддержка среды разработки Java-приложений, включая прямую и обратную генерацию классов Java формата JAR, а также работу с файлами формата CAB и ZIP.

## 3.6 Средства создания пользовательского интерфейса

HTML – стандартизированный язык гипертекстовой разметки документов для создания веб-страниц и веб-приложений [17]. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS (каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием HTML [18]. CSS предназначен для разделения представления и содержимого, включая макеты, цвета и шрифты.

Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения [19].

Основные инструменты Bootstrap:

– Сетки – заранее заданные размеры колонок, которые можно сразу же использовать, например, ширина колонки 140 px относится к классу .span2 (.col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа;

– Шаблоны – фиксированный или резиновый шаблон документа;

– Типографика – описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.;

– Медиа – предоставляет некоторое управление изображениями и видео;

– Таблицы – средства оформления таблиц, вплоть до добавления функциональности сортировки;

– Формы – классы для оформления форм и некоторых событий, происходящих с ними;

– Навигация – классы оформления для панелей, вкладок, перехода по страницам, меню и панели инструментов;

– Алерты – оформление диалоговых окон, подсказок и всплывающих окон.

# 4 Проектирование И разработка программного продукта

## 4.1 Разработка диаграммы вариантов использования

В рамках эксплуатации программного продукта преподаватель столкнется с необходимостью добавлять, создавать, изменять, удалять, сохранять, просматривать аккаунты студентов, а также списки групп. Еще преподавателю необходим инструмент для создания и модернизации курса обучения.

Студентам нужно предоставить возможность просмотра материала, выполнение тестов, выполнение практических, лабораторных и контрольных работ, а также возможность отправки отчетности о выполненной работе.

Для более наглядного представления о возможностях системы, я разработал диаграмму вариантов использования, которая представлена на рисунке 4.1.

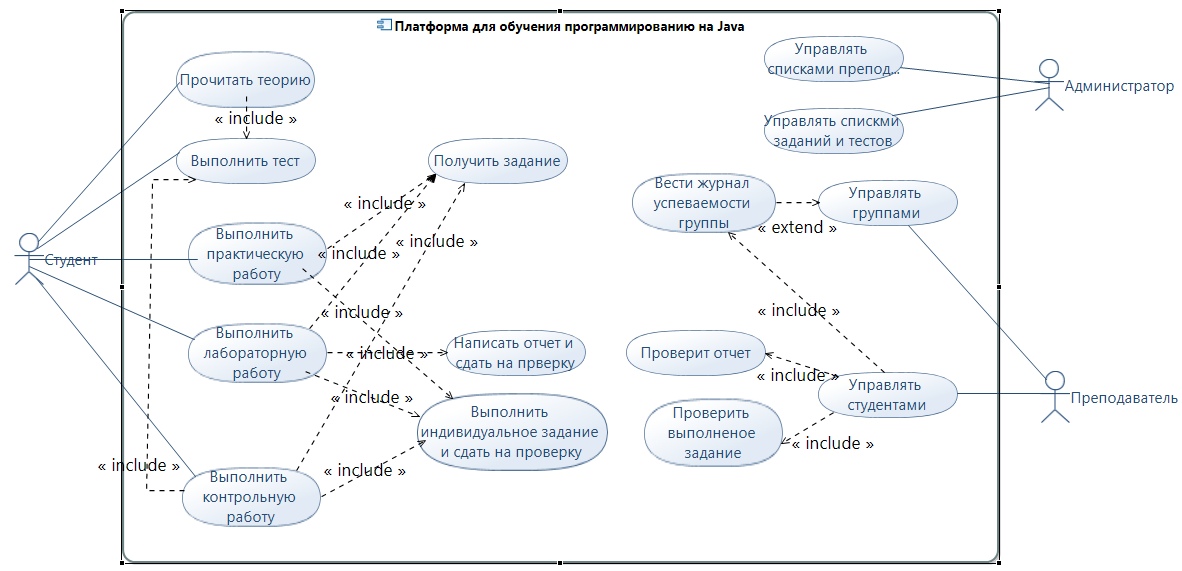


Рисунок 4.1 – Диаграмму вариантов использования

## 4.2 Проектирование и разработка модели хранения данных

При анализе предметной области, я выявил следующие сущности, которые учувствуют в бизнес-модели предметной области:

– Участники: преподаватели, студенты;

– Материалы: теоретический материал, практический материал, тесты;

– Документы: отчеты по лабораторным работам, результаты практической работы, курсовые проекты, контрольные работы, журнал успеваемости;

На основе сущностей предметной области я разработал список сущностей, которые буду учувствовать в разрабатываемой системе для хранения данных:

– Пользователи: обычные пользователи, преподаватели, администраторы;

– Теоретический и практический материалы: курсы, темы, разделы, вопросы, ответы, тесты, задания;

– Документы: отчеты по лабораторным работам, результаты практической работы, оценки;

На основе данных сущностей я разработал диаграмму сущностей представленную на рисунке 4.2.

Для создания модели данных я использовал встроенный инструмент Spring Boot, а именно Spring JPA – спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных [20].

Реализует JPA библиотека Hibernat – самая популярная реализация спецификации JPA, предназначенная для решения задач объектно-реляционного отображения (ORM) [21].

ORM (Object-Relational Mapping, объектно-реляционное отображение) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных» [22].

Схема преобразования POJO объектов (простой Java-объект) в таблицы базы данных показана на рисунке 4.3.

Результат преобразования сущностей в таблицы базы данных представлен на рисунке 4.4.

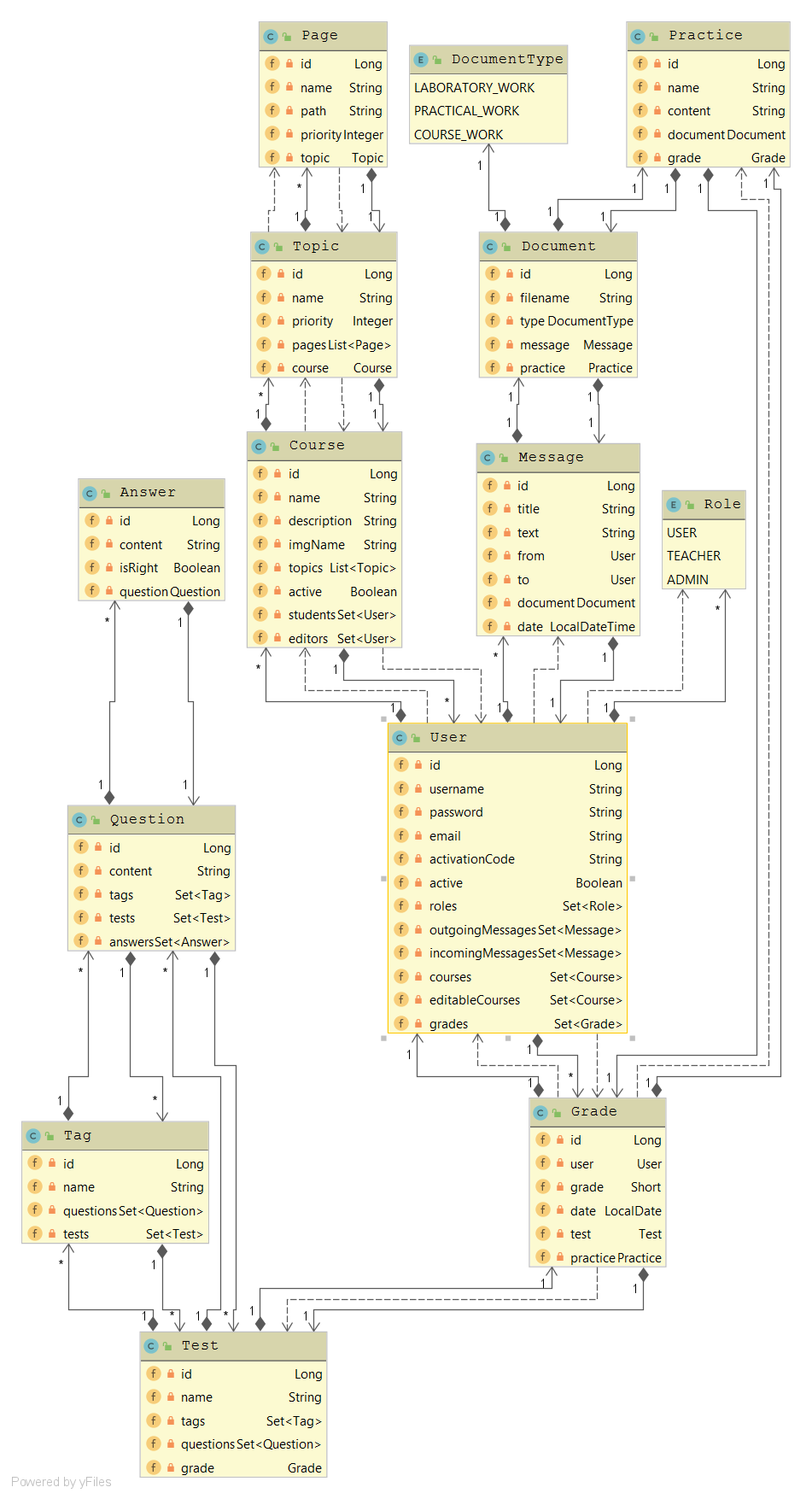


Рисунок 4.2 – Диаграмма сущностей

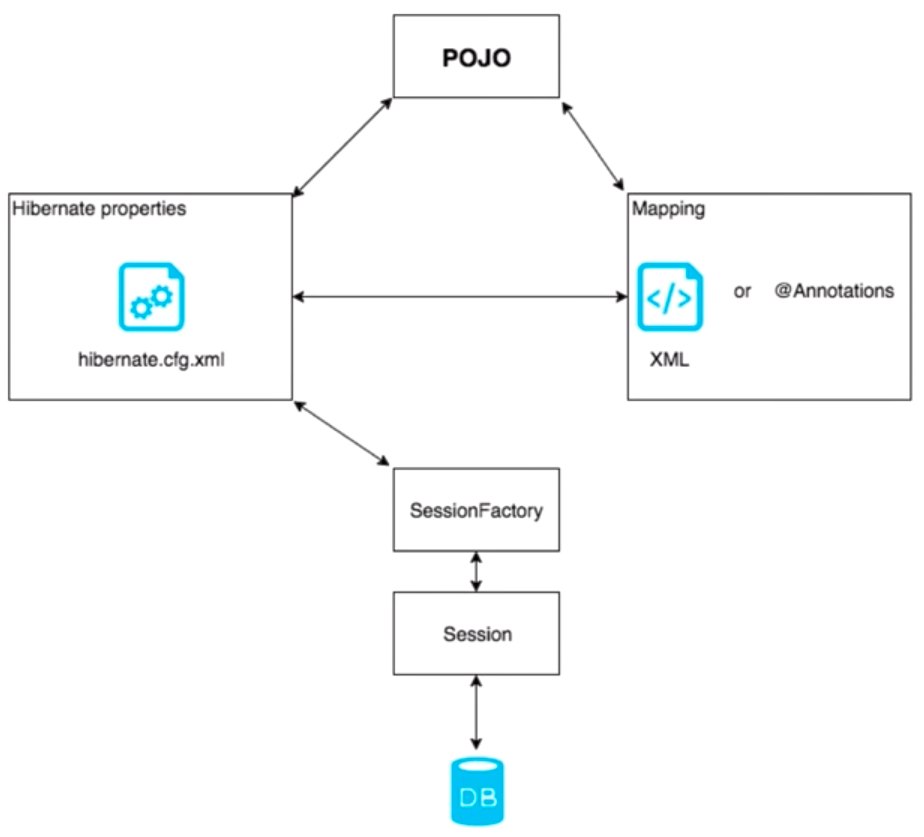


Рисунок 4.3 – Схема преобразования Java-объектов в таблицы базы данных

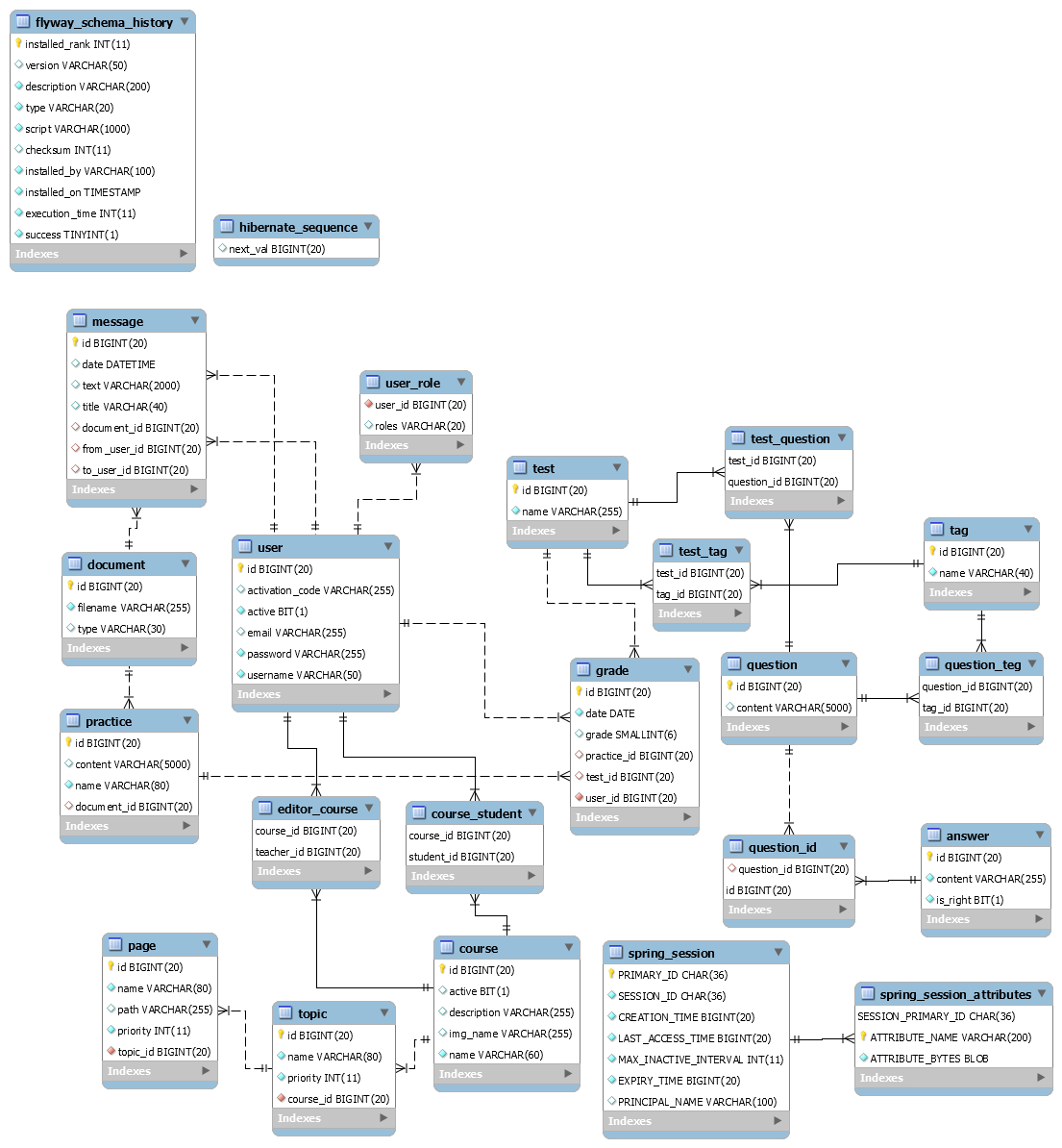


Рисунок 4.4 – Модель базы данных

## 4.3 Разработка структуры проекта

**4.3.1** Spring MVC.

Так как система строиться «клиент-серверной» архитектуре, я воспользуюсь реализацией Spring Web MVC – это оригинальный веб-фреймворк, основанный на Servlet API, который был включен в Spring Framework с самого начала [23].

Структура Spring Web MVC представлена на рисунке 4.5.

DispatcherServlet – это главный контроллер (Front controller), который обрабатывает все входящие и все исходящие запросы. Все поступившие запросы он передает на обработку HandlerMapping контроллеру, который должен решить какому, из все созданных разработчиком контроллеров должен делегировать этот запрос для выполнения запрашиваемой пользователем операции.

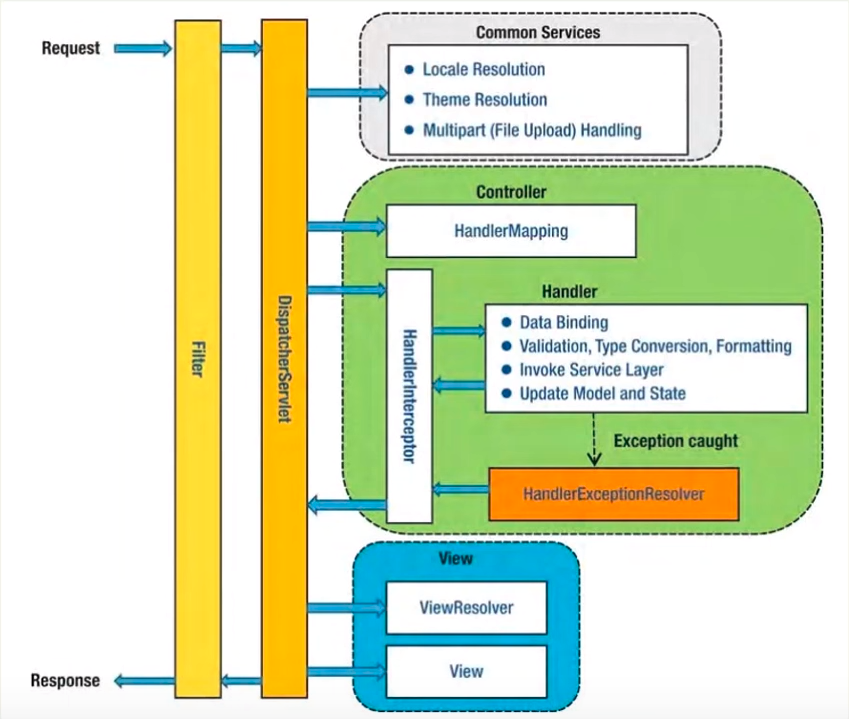


Рисунок 4.5 – Структура Spring Web MVC

После HandlerMapping дает команду главному контроллеру (DispatcherServlet), чтобы тот передал запрос нужному контроллеру. Контроллер выполняет операцию, создает модель (Model), где храниться данные для пользователя и имя представления (View Name). Контроллер передает главному контроллеру данные (ModelAndView), достает View Name, передает контроллеру представлений (ViewResolver), который находит нужное представление (View) – шаблон, на основе которого будет строиться ответ пользователю, для построения ответа пользователю, отдает нужное представление главному контроллеру. Контроллер на основе представления (View) и данных (Model) строит ответ пользователю и отправляет его.

**4.3.2** Структура разрабатываемого проекта

Структура проекта представлена на рисунке 4.6.

Исходя из требований к разрабатываемой системе и выбранной архитектуры приложения, я спроектировал следующую структуру проекта:

– Контроллеры (controller), которые выполняют обработку запросов пользователей и выполняют требуемые операций. Диаграмма классов-контроллеров проекта представлена на рисунке 4.7;

– Сущности (entity), которые занимаются хранением данных в удобной форме. Диаграмма классов-сущностей представлена на рисунке 4.2;

– Сервисы (service), которым делегируется выполнение сложных операция запрашиваемые пользователем. Диаграмма классов-сервисов представлена на рисунке 4.8;

– DAO (Data Access Object) слой, который выполняет CRUD-операции для получения доступа к хранимым данным в базе данных. Диаграмма классов-dao представлена на рисунке 4.9;

– Представления (templates), которые предназначены для построения ответа пользователю;

– Настройки окружения проекта (config, application.properies) – настраивает проект для работы приложения. Диаграмма классов-кофигурации представлена на рисунке 4.10.

Диаграмма пакетов представлена на рисунке 4.11.

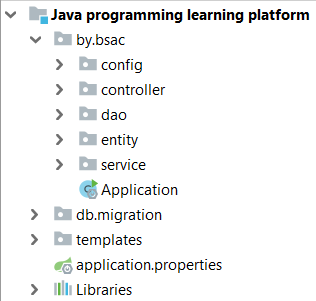


Рисунок 3.6 – Структура разрабатываемого проекта

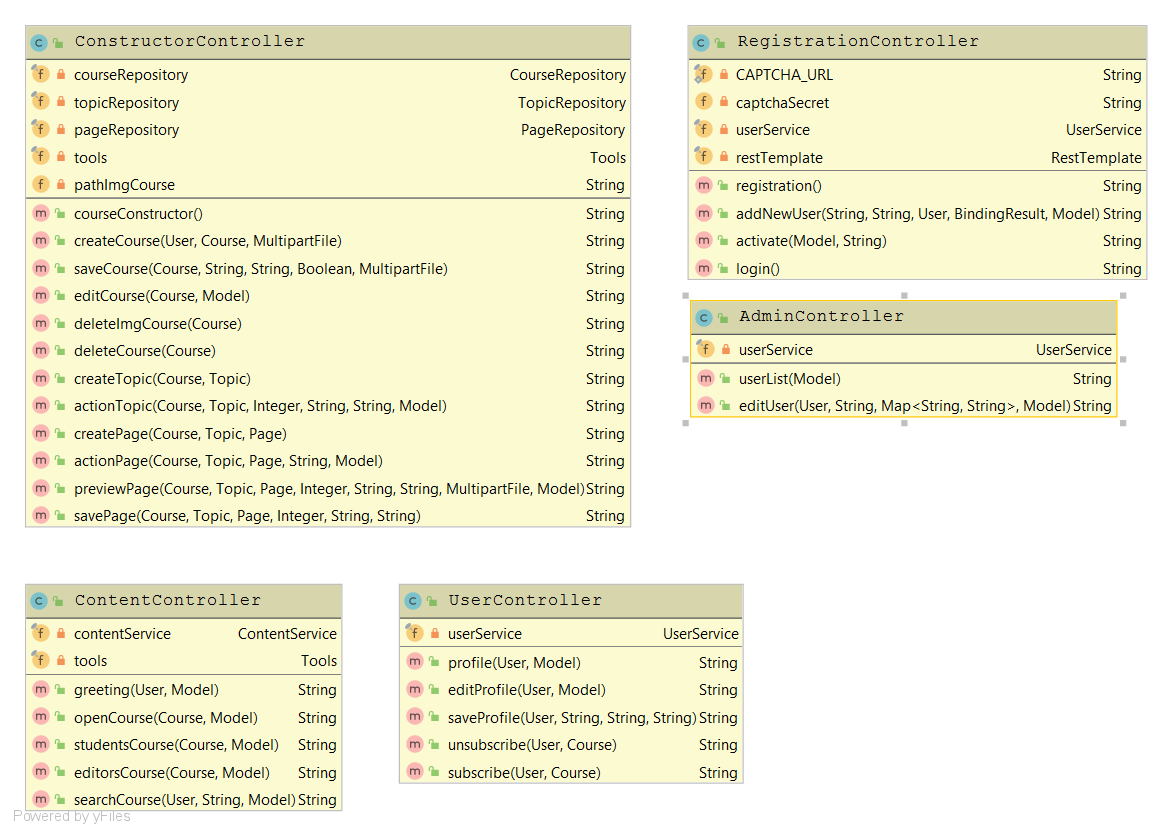


Рисунок 4.7 – Диаграмма классов-контроллеров

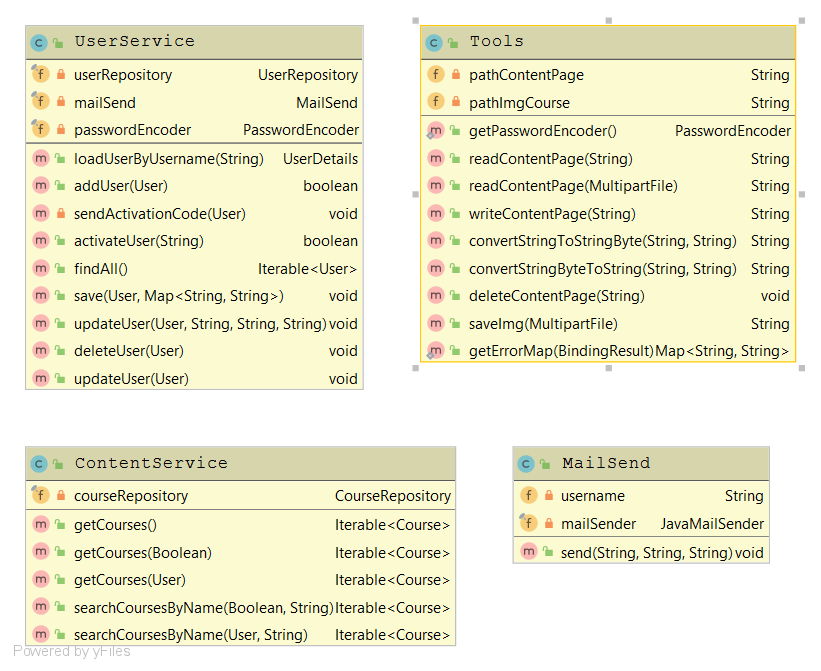


Рисунок 4.8 – Диаграмма классов-сервисов

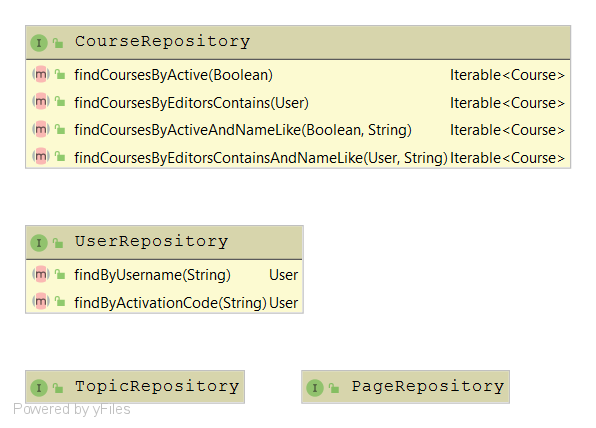


Рисунок 4.9 – Диаграмма классов-dao

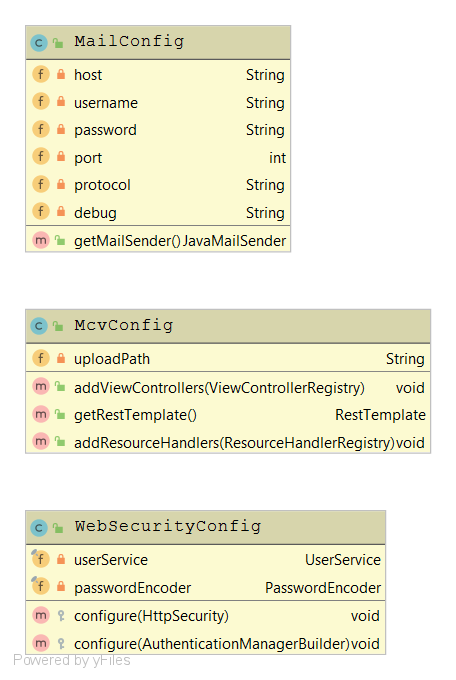


Рисунок 4.10 – Диаграмма классов-конфигурации

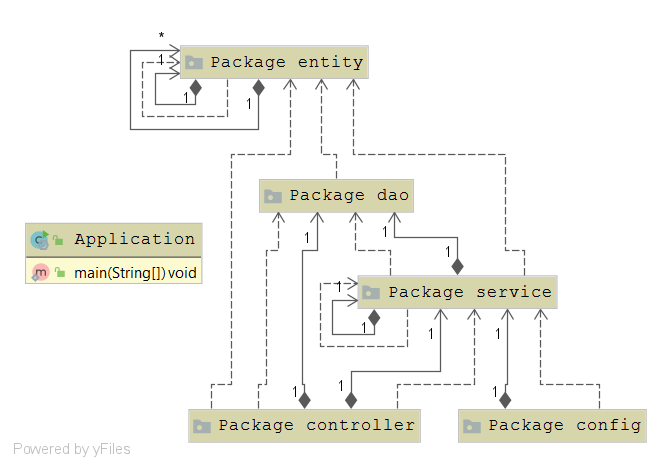


Рисунок 4.11 – Диаграмма пакетов

**4.3.3** Spring Dependency Injection

В основе работы Spring лежит 2 термина: инверсия управления и внедрение зависимостей.

# заключение

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 УО Белорусская государственная академия связи [Электронный ресурс]. – Режим доступа: <http://bsac.by/> – Дата доступа: 16.05.2019.

2 Рябычина, О. П. Конструирование программ и языки программирования. Учебная программа учреждения высшего образования по учебной дисциплине для специальности: 1-45 01 02 Инфокоммуникационные системы (по направлениям) по направлению: 1-45 01 02-03 Инфокоммуникационные системы (сопровождение программного обеспечения) / О. П. Рябычина. – Минск : УО БГАС, 2015. – 13 с.

3 ГОСТ Р ИСО/МЭК ТО 12182 – 2002. Информационная технология. Классификация программных средств.

4 Бахтизин В. В. Технология разработки программного обеспечения / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010. – 267 с.

5 Фатрелл, Р. Управление программными проектами: достижение оптимального качества при минимуме затрат / Р. Фатрелл, Д. Шафер, Л. Шафер. – М. : Вильямс, 2003.

6 Каменнова, М. С. Системный подход к проектированию сложных систем / М. С. Каменнова // Журнал д-ра Добба. – 1993. – №1.

7 IEEE Std. 1320.2–1998. Стандарт IEEE по синтаксису и семантике языка концептуального моделирования IDEFIX97 (IDEF Object). – Введ. 1998-06-25. – Нью-Йорк : IEEE, 1998.

8 Классификация требований к ПО и ее представление в стандартах и методологиях [Электронный ресурс]. – Режим доступа: <http://2006.secrus.org/upload/files/63.pdf> – Дата доступа: 19.05.2019.

9 Write once, run anywhere? [Электронный ресурс]. – Режим доступа: <https://www.computerweekly.com/feature/Write-once-run-anywhere> – Дата доступа: 20.05.2019.

10 The Java Language Environment [Электронный ресурс]. – Режим доступа: <https://www.oracle.com/technetwork/java/intro-141325.html> – Дата доступа: 20.05.2019.

11 Spring Framework [Электронный ресурс]. – Режим доступа: <https://spring.io/projects/spring-framework> – Дата доступа: 20.05.2019.

12 Spring Boot [Электронный ресурс]. – Режим доступа: <https://spring.io/projects/spring-boot> – Дата доступа: 20.05.2019.

13 IntelliJ IDEA [Электронный ресурс]. – Режим доступа: <https://jetbrains.ru/products/idea/> – Дата доступа: 20.05.2019.

14 What is MySQL? [Электронный ресурс]. – Режим доступа: <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html> – Дата доступа: 20.05.2019.

15 Introduction to Java's Architecture [Электронный ресурс]. – Режим доступа: <https://www.artima.com/insidejvm/ed2/introarch.html> – Дата доступа: 20.05.2019.

16 Apache Tomcat [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Apache\_Tomcat – Дата доступа: 20.05.2019.

17 HTML [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/HTML> – Дата доступа: 20.05.2019.

18 CSS [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Cascading_Style_Sheets> – Дата доступа: 20.05.2019.

19 Bootstrap [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Bootstrap\_(front-end\_framework) – Дата доступа: 20.05.2019.

20 Java Persistence API FAQ [Электронный ресурс]. – Режим доступа: https://www.oracle.com/technetwork/java/javaee/persistence-jsp-136066.html – Дата доступа: 21.05.2019.

21 Hibernate (framework) [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Hibernate\_(framework) – Дата доступа: 21.05.2019.

22 ORM [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ORM – Дата доступа: 21.05.2019.

23 Web on Servlet Stack [Электронный ресурс]. – Режим доступа: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html – Дата доступа: 21.05.2019.

**ПРИЛОЖЕНИЕ А**

Таблица А.1 – Выбор модели жизненного цикла на основе характеристик требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии  категории  требований | Каскадная | V-образная | RAD | Инкре-  ментная | Быстрого  прототипи-  рования | Эволюци-  онная |
| 1 | Являются ли требования к  проекту легко определи-  мыми и реализуемыми? | Да | Да | Да | Нет | Нет | Нет |
| 2 | Могут ли требования быть  сформулированы в начале  ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3 | Часто ли будут изменяться  требования на протяжении  ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |
| 4 | Нужно ли демонстрировать  требования с целью их оп-  ределения? | Нет | Нет | Да | Нет | Да | Да |
| 5 | Требуется ли проверка  концепции программного  средства или системы? | Нет | Нет | Да | Нет | Да | Да |
| 6 | Будут ли требования изме-  няться или уточняться с  ростом сложности системы  (программного средства) в  ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7 | Нужно ли реализовать ос-  новные требования на ран-  них этапах разработки? | Нет | Нет | Да | Да | Да | Да |

Таблица А.2 – Выбор модели жизненного цикла на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии  категории  требований | Каскадная | V-образная | RAD | Инкре-  ментная | Быстрого  прототипи-  рования | Эволюци-  онная |
| 1 | Являются ли проблемы  предметной области проек-  та новыми для большинст-  ва разработчиков? | Нет | Нет | Нет | Нет | Да | Да |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии  категории  требований | Каскадная | V-образная | RAD | Инкре-  ментная | Быстрого  прототипи-  рования | Эволюци-  онная |
| 2 | Являются ли инструмен-  тальные средства, исполь-  зуемые в проекте, новыми  для большинства разработ-  чиков? | Да | Да | Нет | Нет | Нет | Да |
| 3 | Изменяются ли роли участ-  ников проекта на протяже-  нии ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4 | Является ли структура про-  цесса разработки более  значимой для разработчи-  ков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5 | Важна ли легкость распре-  деления человеческих ре-  сурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6 | Приемлет ли команда раз-  работчиков оценки, про-  верки, стадии разработки? | Да | Да | Нет | Да | Да | Да |

Таблица А.3 – Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии  категории  требований | Каскадная | V-образная | RAD | Инкре-  ментная | Быстрого  прототипи-  рования | Эволюци-  онная |
| 1 | Будет ли присутствие поль-  зователей ограничено в ЖЦ  разработки? | Да | Да | Нет | Да | Нет | Нет |
| 2 | Будут ли пользователи оце-  нивать текущее состояние  программного продукта  (системы) в процессе раз-  работки? | Нет | Нет | Нет | Да | Да | Да |
| 3 | Будут ли пользователи во-  влечены во все фазы ЖЦ  разработки? | Нет | Нет | Да | Нет | Да | Нет |
| 4 | Будет ли заказчик отслежи-  вать ход выполнения про-  екта? | Нет | Нет | Нет | Нет | Да | Да |

Таблица А.4 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии  категории  требований | Каскадная | V-образная | RAD | Инкре-  ментная | Быстрого  прототипи-  рования | Эволюци-  онная |
| 1 | Разрабатывается ли в про-  екте продукт нового для ор-  ганизации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2 | Будет ли проект являться  расширением существую-  щей системы? | Да | Да | Да | Да | Нет | Нет |
| 3 | Будет ли проект крупно-  или среднемасштабным? | Нет | Нет | Нет | Да | Да | Да |
| 4 | Ожидается ли длительная  эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |
| 5 | Необходим ли высокий  уровень надежности про-  дукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6 | Предполагается ли эволю-  ция продукта проекта в те-  чение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7 | Велика ли вероятность из-  менения системы (продук-  та) на этапе сопровожде-  ния? | Нет | Нет | Нет | Да | Да | Да |
| 8 | Является ли график сжа-  тым? | Нет | Нет | Да | Да | Да | Да |
| 9 | Предполагается ли повтор-  ное использование компо-  нентов? | Нет | Нет | Да | Да | Да | Да |
| 10 | Являются ли достаточными  ресурсы (время, деньги, ин-  струменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |

**ПРИЛОЖЕНИЕ Б**

