Московский физико-технический институт (государственный университет)

Задание №2 по курсу

«Проектирование Программных Систем»

<ИнПлан>

Выполнили:
Давыдов Виталий 193
Синотов Илья 191

Москва
2013

Содержание

1.Постановка задачи

1.1.Состав второго задания

Суть второго задания состоит в реализации модели задания №1 на выбранной платформе. Платформа включает в себя набор классов, реализующий функции доступа к базам данных, коммуникации по сети, рисование UI и решающие другие вспомогательные задачи. Для каждого проекта классы, входящие в платформу оговариваются отдельно. Обратитесь к своему семинаристу для согласования списка классов платформы реализации.

Задание включает:

Задание 1 (модель анализа, описание вариантов использования, диаграммы, обоснование)

Модель реализации - диаграммы вариантов использования, диаграммы классов

Интерпретация ключевых взаимодействий/конечных автоматов динамической модели (2-3 штуки)

Обоснование применения шаблонов проектирования и других проектировочных решений

Модель реализации должна включать отдельную диаграмму (или несколько, если требуется) классов (или внутренней структуры), на которой описано, каким образом реализована модель предметной области, используя библиотеки и классы выбранной платформы.

В дополнение к первому заданию, второе должно включать:

описания основного и альтернативных сценариев в каждом варианте использования

диаграмма классов реализации, в том числе их взаимодействия с платформой / выбранным каркасом

(при необходимости) диаграммы конечных автоматов для описания жизненного цикла классов реализации

диаграммы кооперации и для каждой из них описание взаимодействия, описывающее реализацию сценариев данного варианта использования

диаграммы деятельности для демонстрации основных сценариев вариантов использования (актеры и boundary классы).

1.2.Описание задачи

Система автоматизации индивидуальных планов. Каждый курс имеет требования, кредиты и относится к направлению. Студенты могут выбирать себе курсы. Для получения специализации нужно прослушать определенный набор базовых курсов по направлениям специализации и несколько дополнительных на общую сумму кредитов.

Все курсы платные, студент может расходовать свой бюджет кредитов на разные курсы. Поэтому должен выбирать на какие курсы ходить, чтобы получить специализацию. Деканат может составлять типовые программы. Профессора могут ставить оценки, студенты отслеживать свою успеваемость, деканат получать отчеты об успеваемости по направлениям и специализациям.

2.Описание предметной области

2.1.Логическая модель бронирования билетов

Данный раздел приводится из задания 1. Привести последнюю версию диаграммы классов.

На рис. 1 приведена диаграмма классов после окончательного анализа всех вариантов использования акторов системы.

Рис. 1. Обновленная диаграмма классов

2.2.Описания и обязанности классов

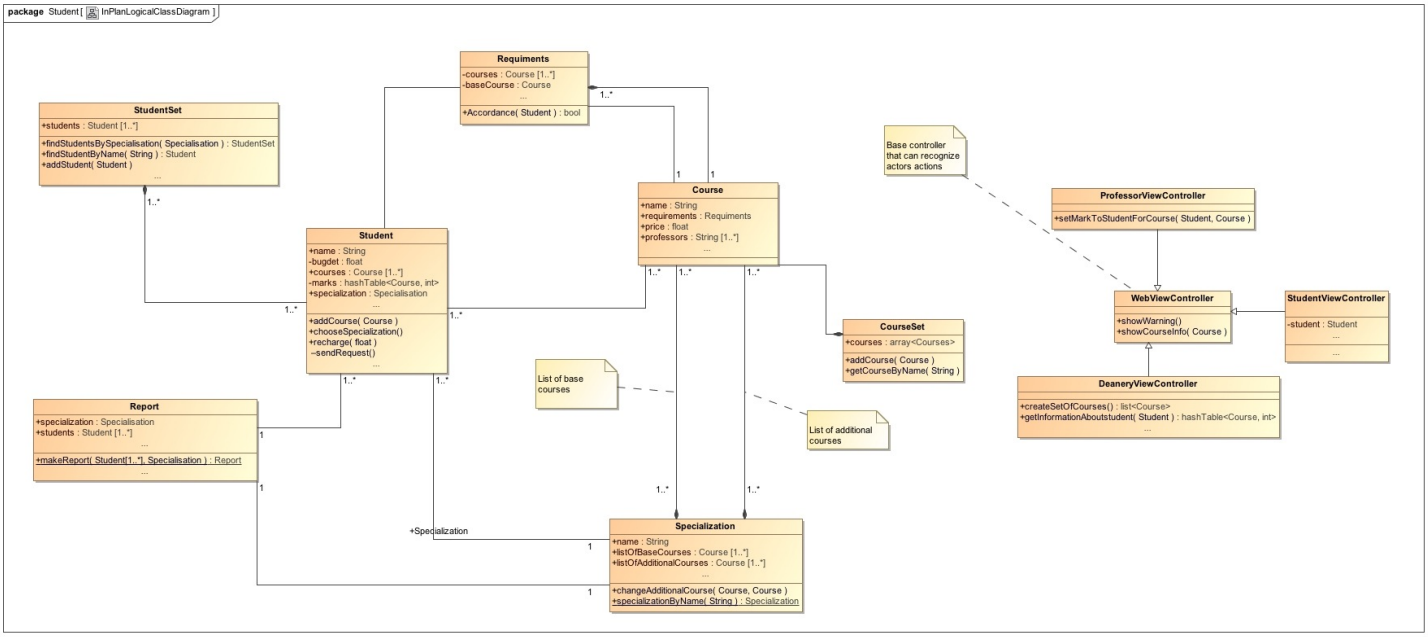
Привести таблицу с перечнем классов и обязанностями, коллегами. Обязанностей и коллеги приводятся из первого задания, агрегированно по всем взаимодействиям.

Таблица 2. Обязанности и коллеги классов предметной области бронирования билетов.

Класс	**Контракт**	**Коллеги**
Student	Хранит информацию о студенте Знает все о студенте Может записывать студента на курс, специализацию Может пополнять баланс в системе	Course
StudentSet	Хранит список студентов	Student
Course	Хранит информацию о курсе	Performance, Requiments
CourseSet	Хранит список курсов	Course
Requiments	Хранит требования для данного курса Проверяет студента на соответствие требованиям	Course, Student
Report	Позволяет сделать отчет по специализации	Student, Specialization
Specialization	Хранию всю информацию о специализации	Course

WebViewControlle r	Хранит базовый интерфейс для остальных классов	
ProfessorViewCon troller	Подкласс WebViewController Предоставляет актору Professor UI	WebViewController
StudentViewContr oller	Подкласс WebViewController Предоставляет актору Student UI	WebViewController
DeaneryViewContr oller	Подкласс WebViewController Предоставляет актору Deanery UI	WebViewController

3.Описание функциональных требований к системе



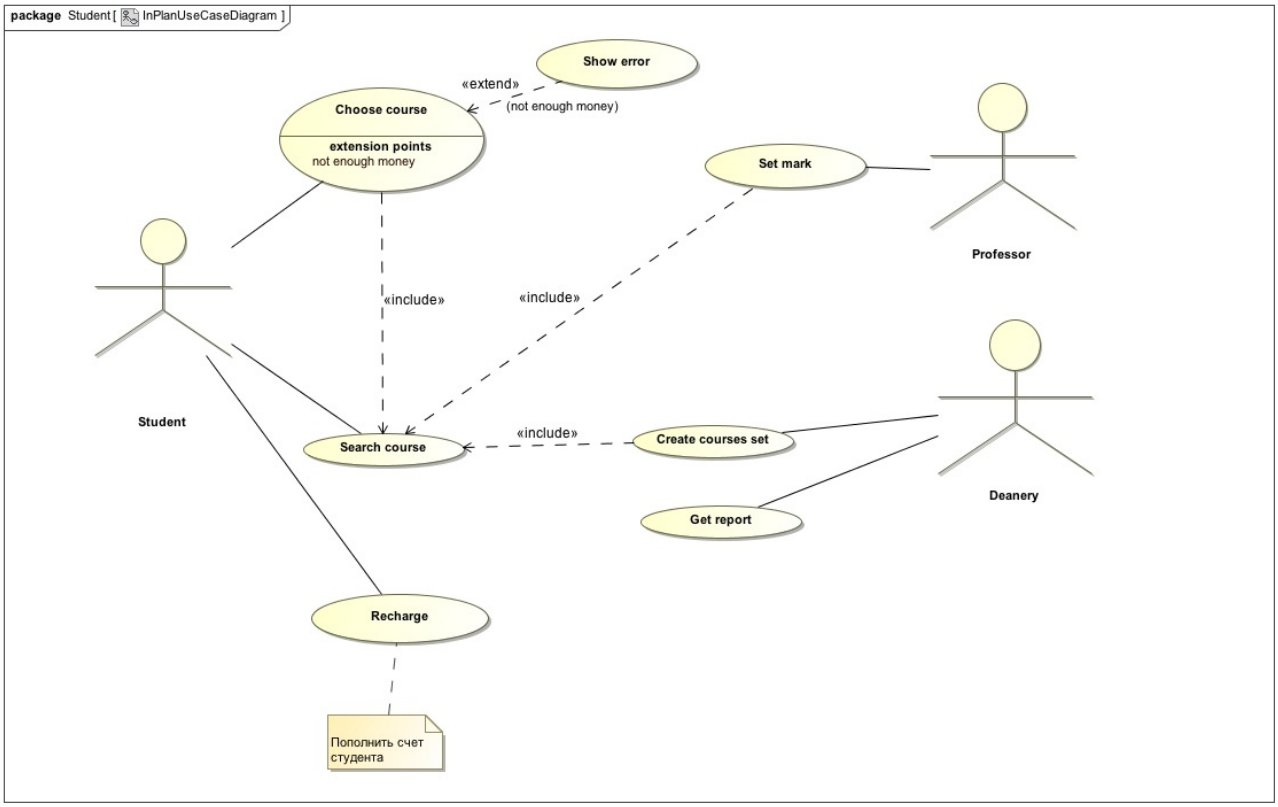
3.1.Модель вариантов использования

Раздел повторяет аналогичный раздел из задания 1. См. пояснения в описании первого задания.

В данной модели акторами являются Student, Professor, Deanery.

Студент может искать курсы и выбирать для себя подходящий. При выборе курса добавлена зависимость <<extend>> так, как у студента может не хватить бюджета на данный курс, после этого будет выдана ошибка.

Профессор может выставлять студентам оценки по своему курсу, здесь добавлена зависимость с <<include>>.



Деканат может получать отчет и составлять набор необходимых курсов.

Рис. 2. Диаграмма вариантов использования

3.1. Вариант использования Set mark.

Акторы: Professor (Профессор)

Цель: Профессор ставит оценку студентам, посещающим его курс.

Предусловия: Профессор ведет курс.

Постусловия: Профессор проставил оценки.

Основной сценарий:

1. Профессор запрашивает список курсов (дополнительно используется вариант использования Search course), которые он ведет. Система показывает список курсов и студентов.
2. Профессор выбирает курс и студента.
3. Ставит ему оценку.

3.1. Вариант использования Search course.

Акторы: Student (Студент), Professor (Профессор), Deanery (Деканат)

Цель: Актор хочет найти курс или просмотреть список всех курсов.

Предусловия: Деканат решил создать набор курсов или профессор собрался выставить оценку студенту по курсу или студент решил найти нужный ему курс.

Постусловия: Актор добился основной своей цели, указанной в предусловиях.

Основной сценарий:

1. Актор запрашивает курс по определенным критериям поиска. Система показывает список курсов.
2. Актор выбирает курс (имеется в виду, что он не платит за него, а получает подробную информацию и описание) и использует в соответствие со своей целью.

2.1.Вариант использования Choose course.

Акторы: Student (Студент)

Цель: Студент хочет записаться на курс.

Предусловия: Студент нашел курс.

Постусловия: Студент записался на курс.

Основной сценарий:

1. Студент производит оплату за курс. Система снимает деньги с бюджета студента и записывает его на участие в курсе.

Альтернативные сценарии:

Если на шаге 1. у Студента не хватает денег, Система уведомляет Студента, и просит его выбрать другой курс.

3.5.Вариант использования Create courses set.

Акторы: Deanery (Деканат)

Цель: Деканат хочет создать набор курсов в рамках специализации.

Предусловия: Существование деканата.

Постусловия: Создан набор курсов и добавлен в список курсов.

Основной сценарий:

1. Деканат составляет список курсов (для этого дополнительно используется вариант использования Search course).

2. Деканат отправляет список. Система добавляет курсы в базу.

2.6.Вариант использования Get report.

Акторы: Deanery (Деканат)

Цель: Деканат хочет получить отчет об успеваемости студентов в рамках направления.

Предусловия: Есть список курсов и студентов.

Постусловия: Отчет создан и получен.

Основной сценарий:

1. Деканат отправляет запрос с информацией о том, по какой специализации получить отчет.
2. Деканат получает отчет.

3.7.Вариант использования Recharge

Акторы: Student (Студент)

Цель: Студент хочет пополнить свой счет (Budget) в системе.

Предусловия: У студента есть банковский счет.

Постусловия: Студент пополнил счет.

Основной сценарий:

1. Отправка запроса в банк на перевод денежных средств.
2. Ожидаем подтверждение из банка о корректности запроса.
3. Увеличивается значение переменной Budget у Student.

Альтернативные сценарии:

Если на шаге 2. не получили подверждение из банка, выводим ошибку.

4.Описание архитектуры системы

4.1.Описание трехуровневой архитектуры

Привести название и описание выбранного архитектурного стиля. Перечень стилей и их описание можно взять у преподавателя.

Система состоит из двух основных компонентов: iOS приложения и Rest сервера. Так же существую как локальная так и серверные базы данных (типа SQL Lite), которые сохраняют все информацию в статическом режиме.

Клиент (iOS приложение) использует дополнительный фреймворк AFNetworking специально для удобного отправления запросов на сервер. Оправка запроса идет в виде JSON'а с необходимыми полями, например,

{

student: {

name:"Vitaly Davydov"

couses : […]

}

}

в ответ так же приходит ответ в виде JSON'а. Парсинг и мэпинг идет в классе-Адаптере над AFNetworking, называемым IWRequester.

Локальная база данных и серверная находятся в постоянной синхронизации (обновление, например, списков курсов на локальной базе данных осуществяется по pull-to-refresh действию пользователя).

Помимо этого, в iOS приложении будет класс IWSession, который показывают текущую сессию пользователя на сервере. По сути, все "общение" пользователя с сервером осуществляется через этот класс.

4.2.Обоснование выбора архитектурного стиля

Привести подробное обоснование, почему был выбран данный архитектурный стиль. Обоснование вида «так было сказано» не является достаточным. Нужно привести рассуждения, основанные на предполагаемом использовании системы, применяемых стилях в схожих системах. Например:

Среди рассмотренных архитектурных стилей:

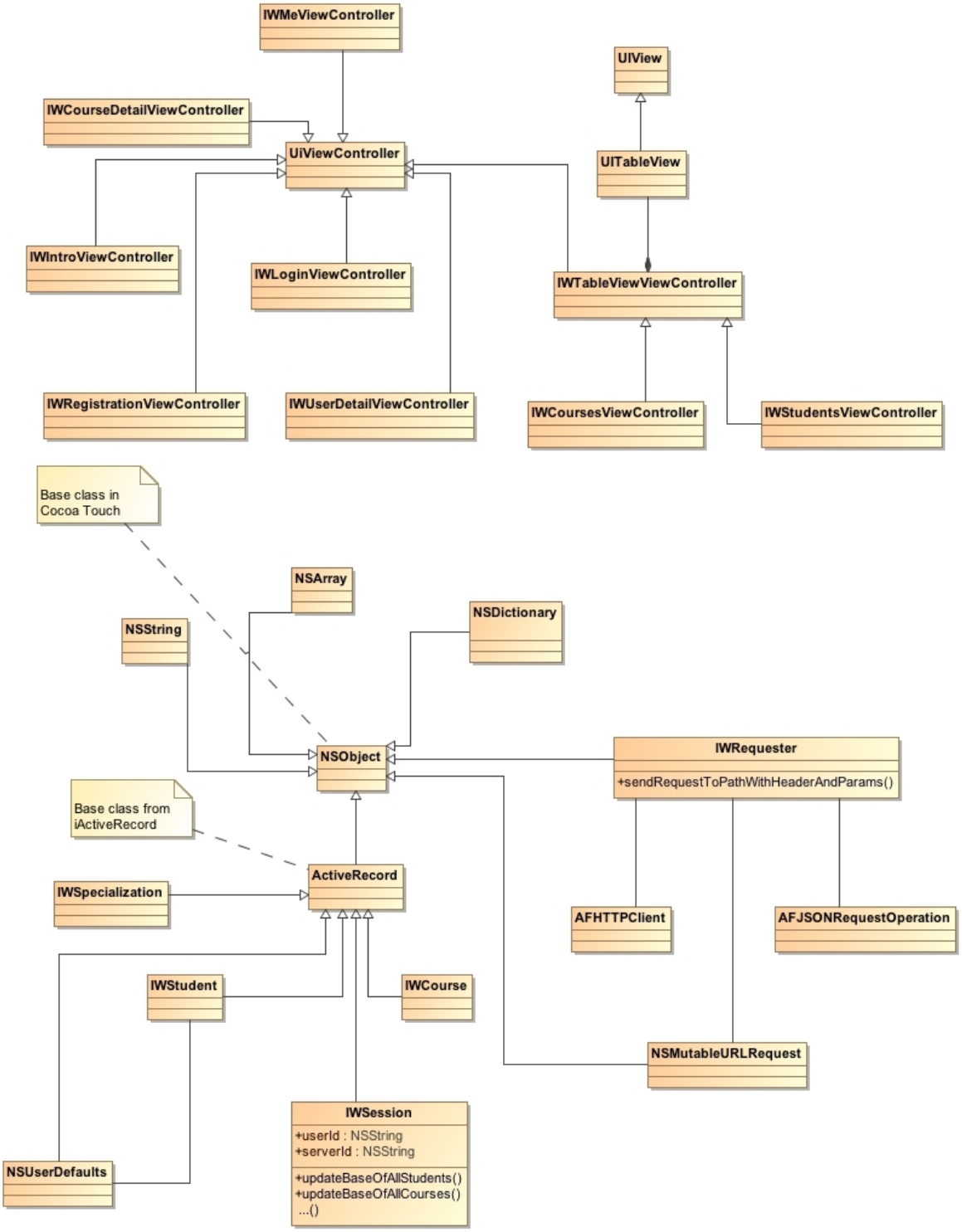
1. трехуровневая архитектура
2. монолитное приложение
3. встраиваемая система
4. …

Для разрабатываемой системы индивидуальных планов обучения, очевидно, подходит трехуровневая архитектура, т.к пользователь в любом случае взаимодейтвует с удаленной системой, с которой синхронизируются остальные пользователи системы.

Остальные стили возможны, но их применение крайне не оптимально в данной ситуации.

4.3.Описание вспомогательных интерфейсов и классов

Рис. 3. Описание классов и интерфейсов платформы реализации



На данных диаграммах представлены описания классов и интерфейсов для платформы для реализации проекта на iOS. Показана иеархия классов от базовых NSObject (для моделей), UIViewController (для контроллеров) и UIView (для представлений), которые являются стандартными вфреймворке CocoaTouch. Так же была использована сторонняя библиотека iActiveRecords (аналог ActiveRecords в RoR) для реализации локальной базы данных SQLite. От нее взят класс ActiveRecord, который является базовым для моделей, хранимых в базе. Классы IWStudent, IWCourse, IWSpecialization и IWRequiments отображают те же классы, что и с записью без IW.

Таблица 2. Обязанности и коллеги классов предметной области бронирования билетов.

Класс	**Контракт**	**Коллеги**
IWIntroViewController	Определяет пользователя системы	IWRegistationViewController
IWRegistationViewController	Регистрирует пользователя в системе	IWStudent
IWLoginViewController	Осуществляет вход пользователя в систему	IWStudent, UITabBarController
IWCoursesViewController	Показывает список всех курсов	IWCourse
IWCourseDetailViewController	Показывает детальную информацию о курсе Предлагает купить его (подписаться на него)	IWCourse, IWStudent
IWMeViewController	Показывает информацию о текущем пользователе	IWStudent
IWStudentsViewController	Показывает всех пользователей в системе	IWStudent
IWRequester	Посылает прямой запрос к серверу Принимает JSON, делает mapping в локальную базу	
IWSession	Показывает сессию текущего пользователя на сервере Обрабатывает ответ с сервера	IWStudent, IWCourse, IWSpecialization, IWRequiments

4.4.Структура компонентов системы и решения по реализации архитектуры

Привести диаграмму компонентов согласно архитектурному стилю. Показать на диаграмме предоставляемые и используемые интерфейсы, соединители. Перечень и назначение компонентов.

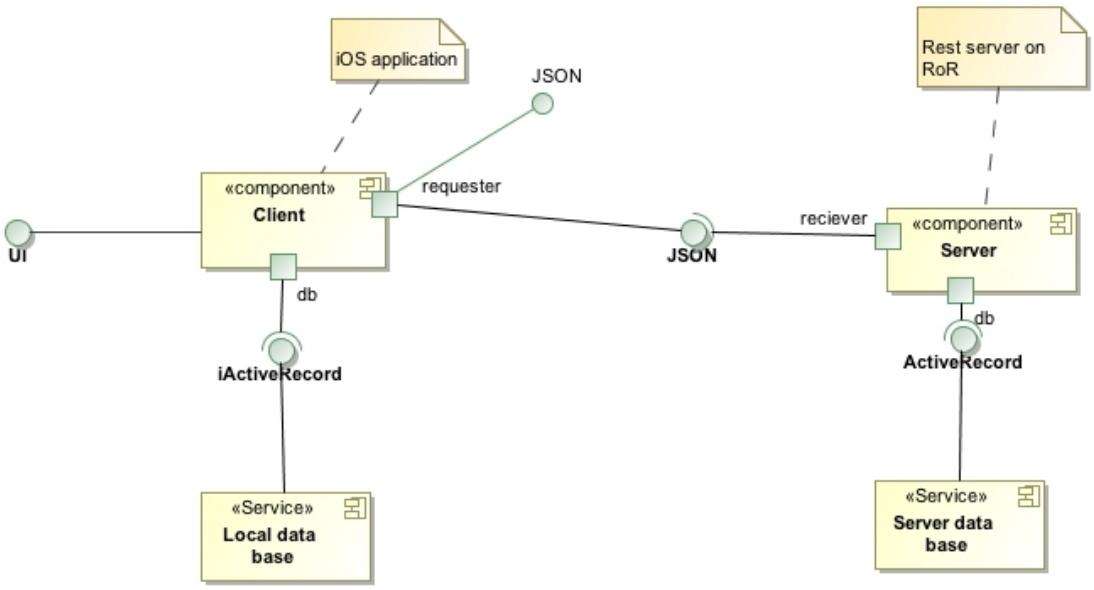


Рис. 3. Структура системы

В таблице указать назначение каждого компонента. Заполняется по описанию архитектуры.

Таблица 2. Обязанности и коллеги классов предметной области бронирования билетов.

Компонент	**Назначение**	**Интерфейсы и порты**
Client	Предоставляет графический интерфейс пользователям. Взаимодействует с сервером Хранит информацию с сервера в локальной бд.	UI requester:~JSON db:~LocalDataBase

Server	Предоставляет HTTP интерфейс, принимает JSON запросы на обработку от клиента. Отправляет JSON ответы. Взаимодействует с базой данных. Содержит сервлеты, реализуемые приложением.	HTTPConnector:HTTP clientUI:ClientUI db:~ServerDataBase reciever:IWReciever
Server data base	Хранит информацию на сервере	Через ActiveRecord
Local data base	Хранит информацию на девайсе пользователя Синхронизируется с сервером	Через iActiveRecord

4.5.Структура пакетов

Привести диаграмму пакетов и зависимостей между ними. Диаграмма пакета должна отражать структуру проекта в среде моделирования.

Привести обоснование распределения элементов модели по пакетам с использованиям принципов проектирования.

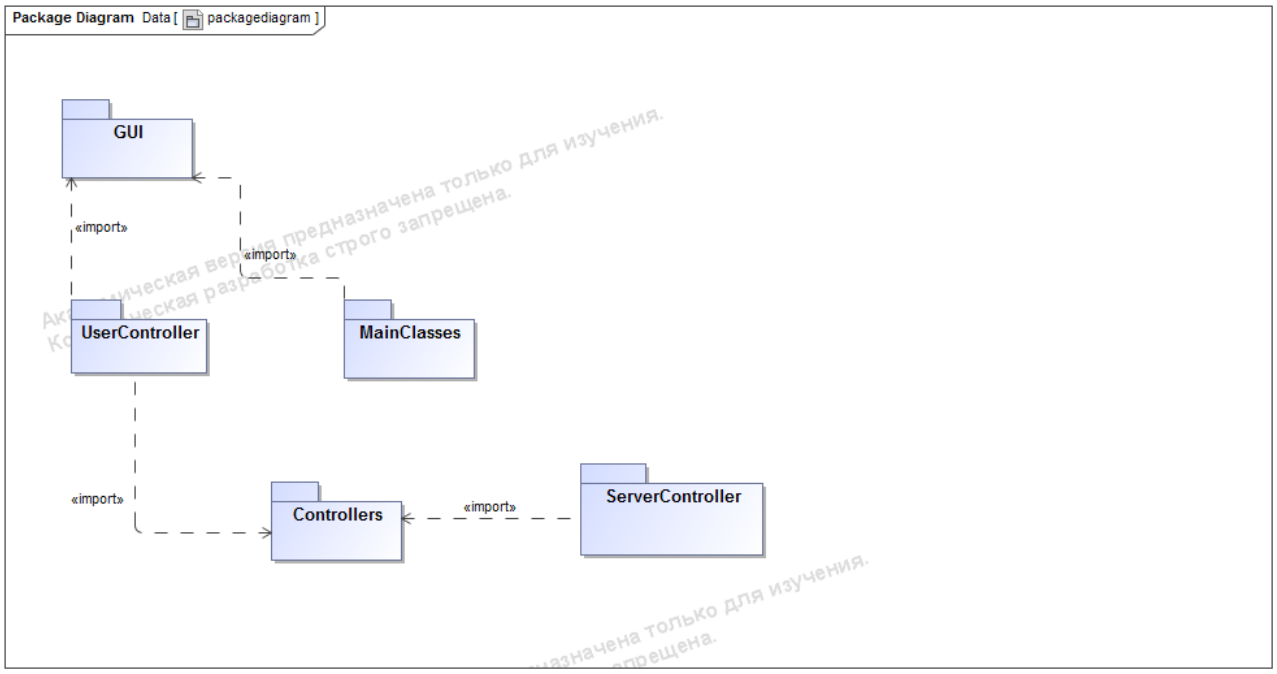


Рис. 4. Структура пакетов модели системы

5.Описание механизмов реализации

В данном разделе по аналогии с первым заданием нужно провести анализ взаимодействий, описанных в вариантах использования, и механизмов инициализации и завершения работы системы. Для них нужно проработать, какие классы будут участвовать в реализации и какие обязанности они будут иметь.

Привести диаграмму классов, на которой показать, как варианты использования реализованы кооперациями. В подразделах рассмотреть каждый ВИ в отдельности, показать сценарий на диаграмме одного из типов:

- *Последовательности для описания интерактивного взаимодействия.*
- *Деятельность для описания координированного использования нескольких объектов, или для описания процесса с несколькими участниками.*
- *Схемы состояний для описания жизненного цикл объекта, описание процесса (через воплощение).*

На основе прототипа iOS приложения произведем детальное описание нескольких вариантов использования.

5.1.Реализация варианта использования Search course

Использовать классы структуры системы из раздела 4.4, классы предметной области или их производные из раздела 2 и новые классы, которые потребуются для назначения обязанностей.

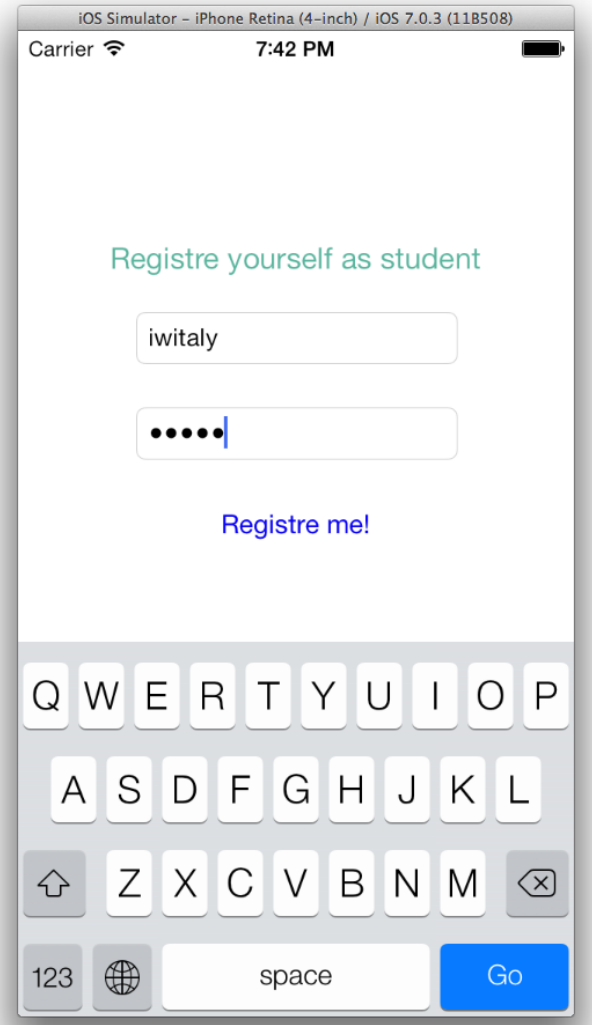
Для основного и альтернативных сценариев выделить и назначить обязанности, затем построить диаграммы поведения: последовательности, схемы состояний или деятельности.

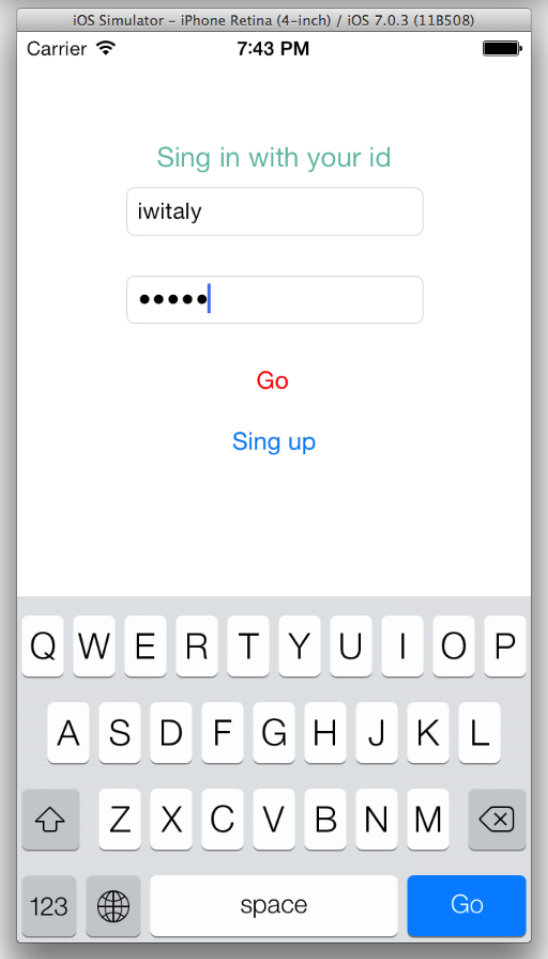
Можно использовать карточки CRC, используя в качестве заготовок модель предметной области и платформу реализации. Для обоснования назначения обязанностей классам следует использовать эвристики GRASP (Information Expert, Creator, Controller, Low Coupling, High Cohesion, Polymorphism, Pure Fabrication, Indirection, Protected Variations) и общие принципы проектирования (OCP, LSP, SRP/ISP, DIP).

Привести таблицу с описанием обязанностей классов для данного сценария. Показать, какие паттерны применяются и каким образом.

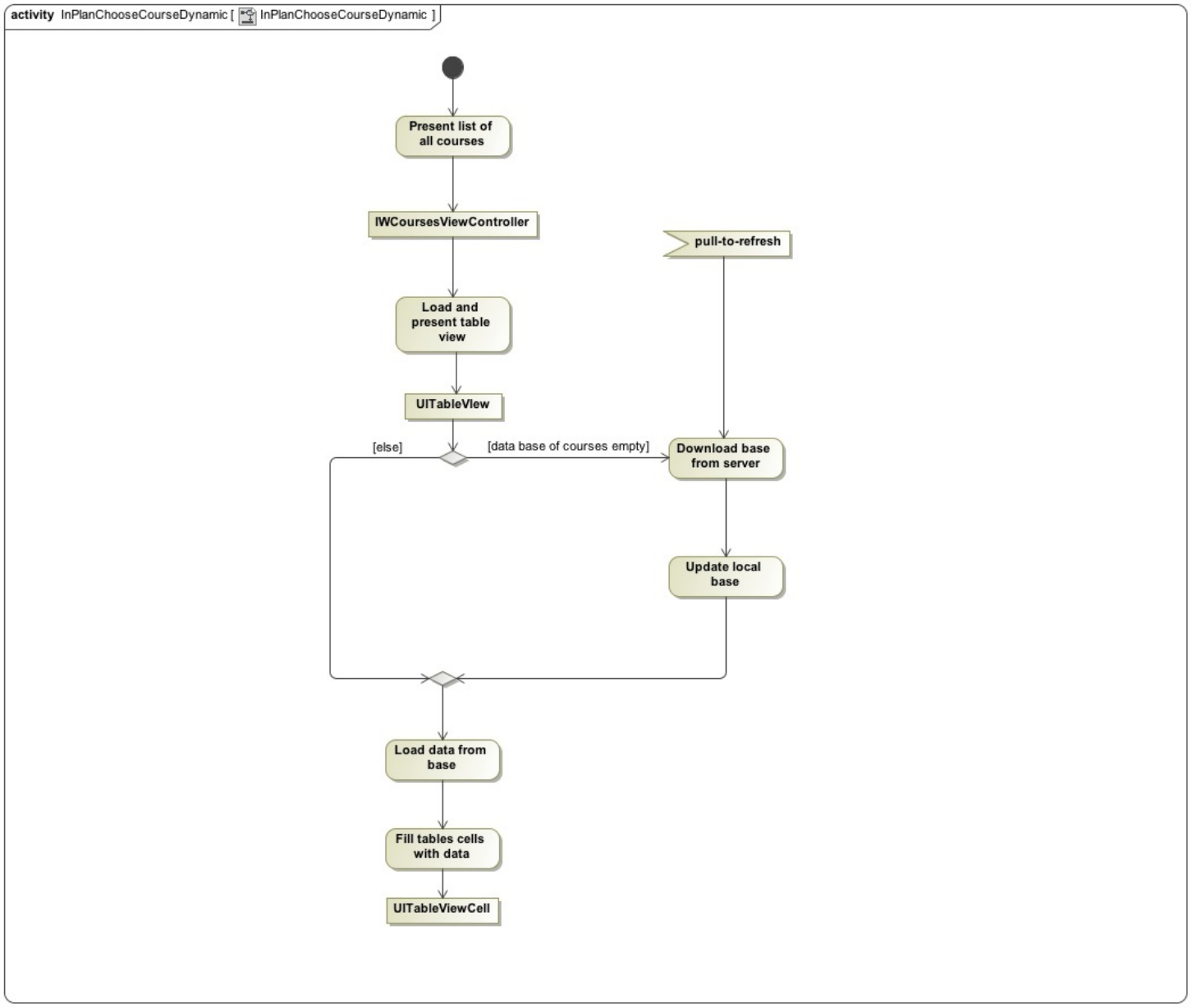
После этого, показать во фрагменте диаграммы классов интерфейсы, ассоциации, атрибуты и операции, реализующие выделенные обязанности. После диаграммы в тексте дать обоснование.

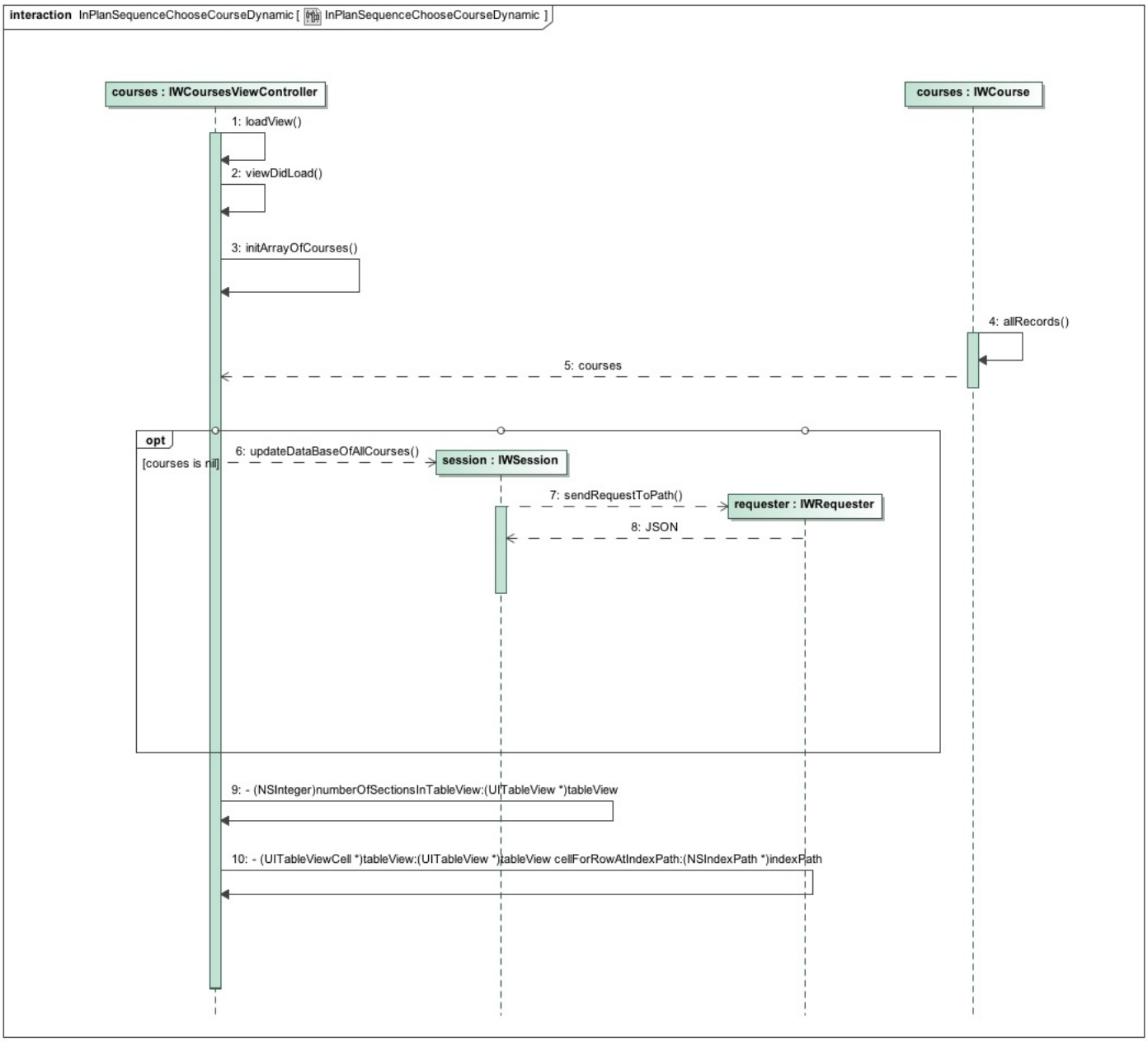
Предполагаем, что пользователь (студент) уже зарегистрировался и залогинен в нашей системе, эти операции реализованны в IWRegistrationViewController и IWLoginViewController, эти процессы выглядят так:


iOS Simulator - iPhone Retina (4-inch) / iOS 7.0.3 (11B508)
Carrier
7:42 PM
Registre yourself as student
iwitaly
Registre me!
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
Go


iOS Simulator - iPhone Retina (4-inch) / iOS 7.0.3 (11B508)
Carrier
7:43 PM
Sing in with your id
iwitaly
Go
Sing up
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
Go

Были построены диаграммы последовательностей и деятельности для данного варианта использования.



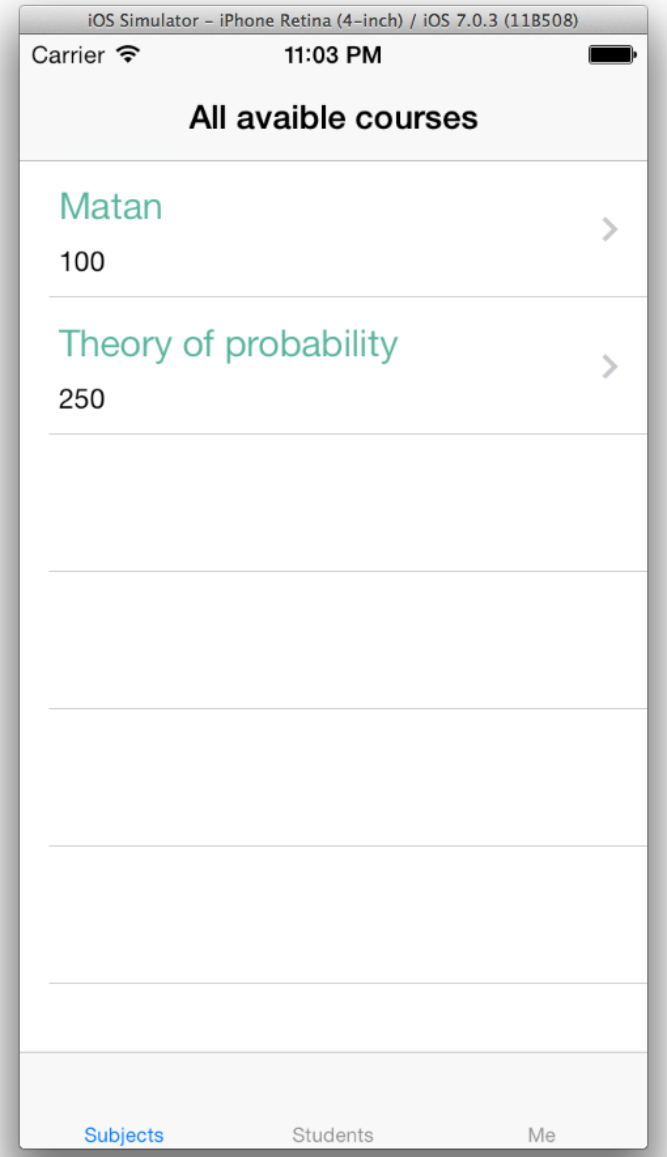


Класс	**Контракт**	**Коллеги**
IWCoursesViewContr oller	Предоставляет интерфейс Выводит список всех курсов	IWCourse, IWSession, UITableView
IWCourse	Хранит список всех курсов	
IWSession	Делегирует обновление локальной базы курсов Обрабатывает notification”	IWRequester, IWCourse
IWRequester	Отправляет запросы на сервер Принимает ответы	

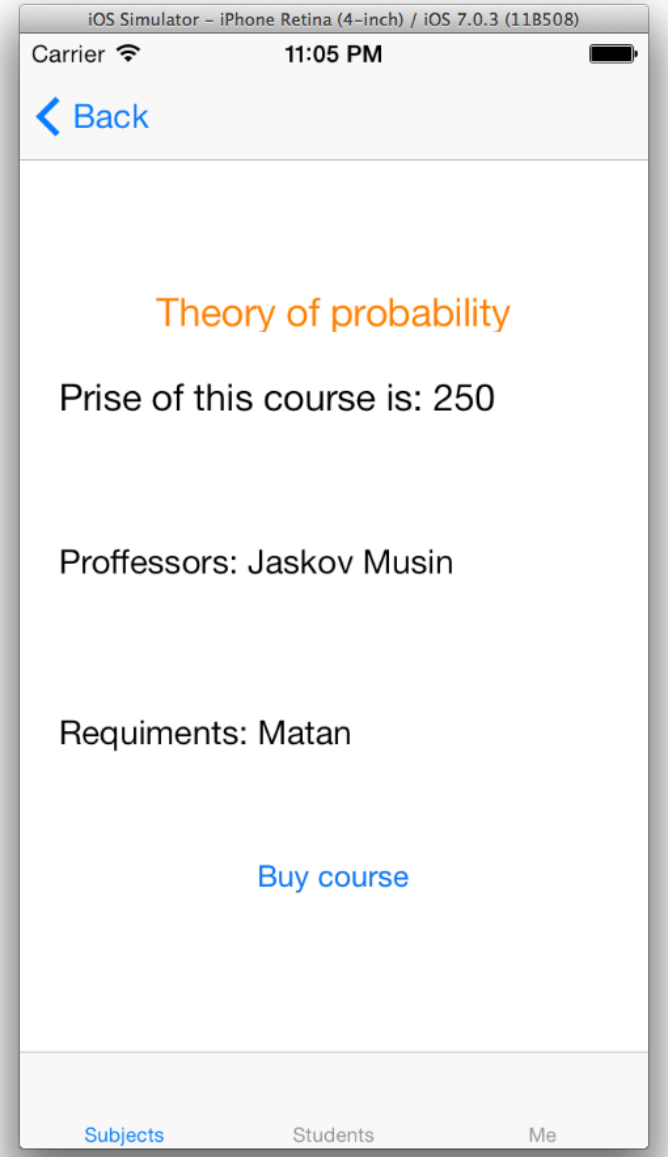
На диаграмме не были указаны “досканально” все методы и классы, между которыми происходит взаимодействие. Например, выставление названия курса и его цены с помощью UILabel было опущено, т.к. не является существенным.

5.2.Реализация варианта использования Choose course

Варианты использования Search course и Choose course не сильно отличаются, в том смысле, что Search course почти полностью использует Choose course.



В этот момент, когда пользователь (в данном случае актор - студент) использует юзкейс Search course. При тапе на ячейку с курсом студент попадает в IWCourseDetailViewController



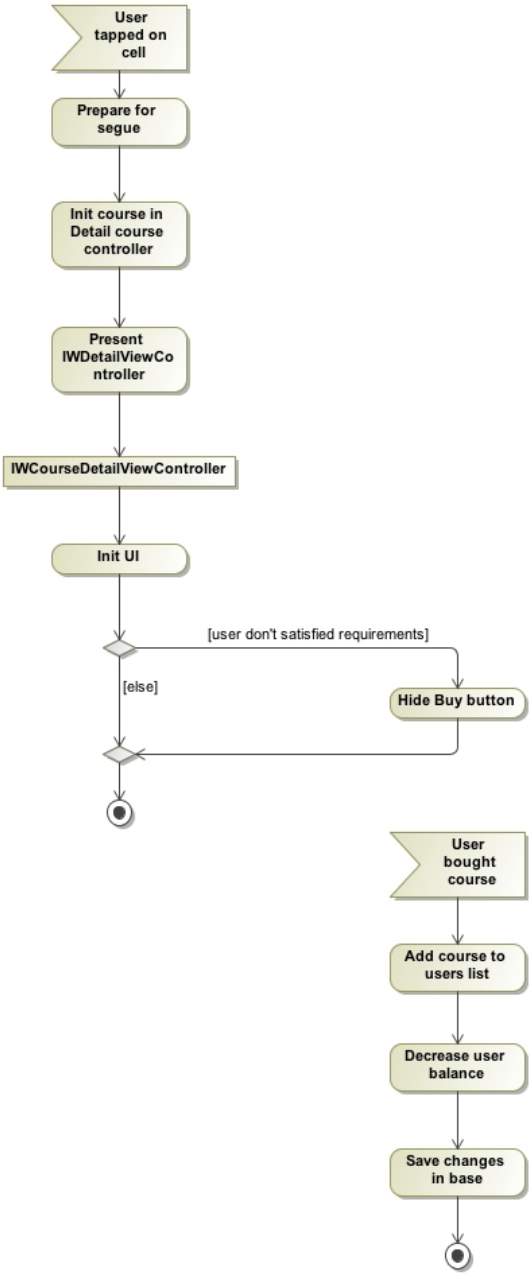
где мы видим информацию о курсе. Так же доступна кнопка для покупки курса. Если студент не удовлетворяет требованиям курса или у него недостаточно денег для покупки курса, кнопка Buy course становится неактивна.

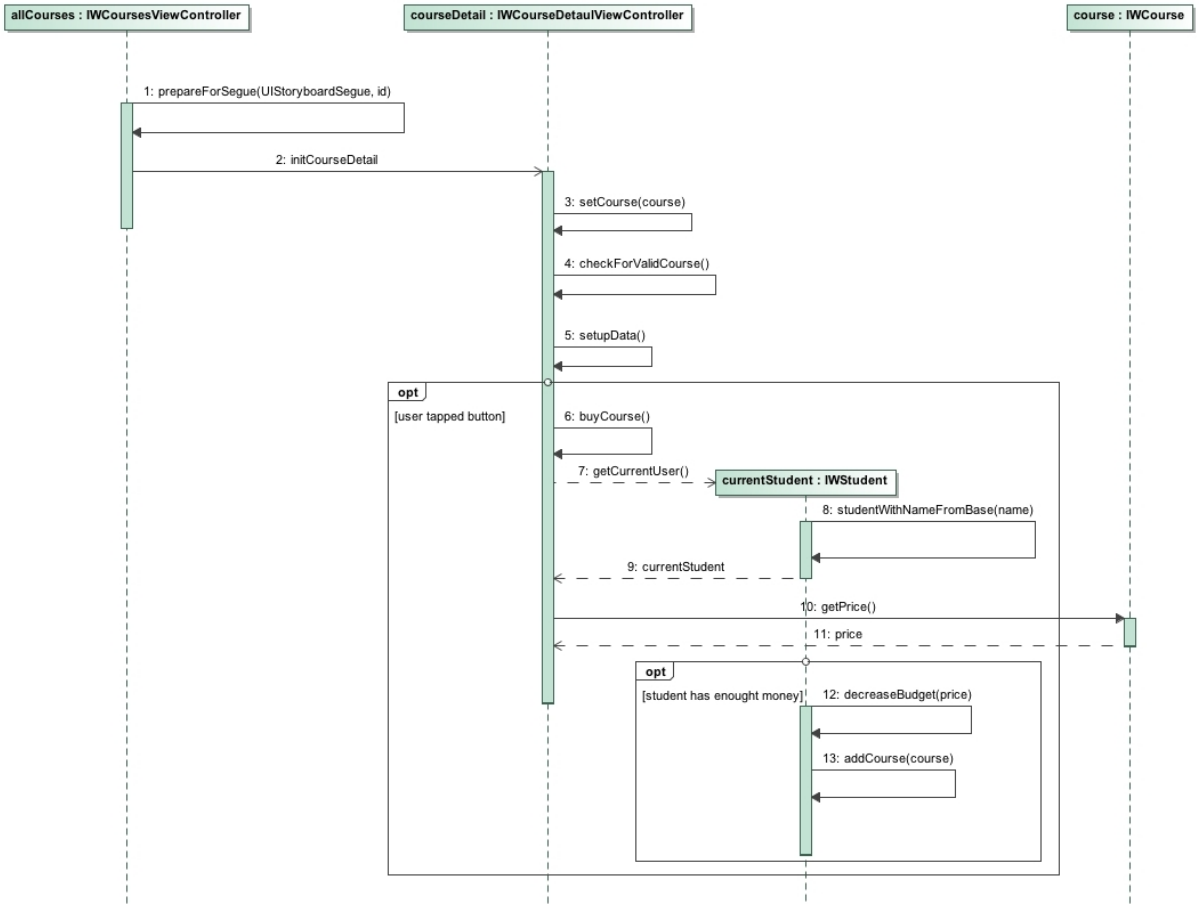
Для навигации используется UINavigationController, который хранит контроллеры в виде стэка. Для перехода используется система Storyboard https://developer.apple.com/library/ios/documentation/general/conceptual/Devpedia-CocoaApp/Storyboard.html.

Диаграммы последовательностей и деятельности: (диаграмма логически начинается с “окончания” деятельности в Choose course, то есть с вызова метода

- (void)performSegueWithIdentifier:(NSString *)identifier sender:(id)sender

который вызывается по событию, когда пользователь нажал на ячейку.)


User tapped on cell
Prepare for segue
Init course in Detail course controller
Present IWDetailViewController
IWCourseDetailViewController
Init UI
[user don't satisfied requirements]
[else]
Hide Buy button
User bought course
Add course to users list
Decrease user balance
Save changes in base



Класс	**Контракт**	**Коллеги**
IWCoursesViewController	Предоставляет интерфейс Выводит список всех курсов Инициализирует курс в Detail view controller Осуществляет Segue на Detail view controller	IWCourse, IWSession, UITableView, IWCourseDetailController
IWCourse	Хранит список всех курсов	
IWSession	Делегирует обновление локальной базы курсов Обрабатывает notification”ы Обеспечивает обновление списка курсов пользователя на сервере	IWRequester, IWCourse
IWRequester	Отправляет запросы на сервер Принимает ответы	
IWCourseDetailViewController	Отображает подробную информацию по курсу Предлагает его купить	IWStudent, IWSession, IWCourse

6.Обеспечение нефункциональных требований

Перечислить в подразделах нефункциональные требования, упоминаемые в постановке задачи. Пояснить, каким образом они удовлетворены в архитектуре и дизайне системы.

6.1.Доступность системы через Интернет

В выбранной трехзвенной архитектуре слои взаимодействуют посредством сетевых протоколов, используемых в Интернет. Поэтому слой представления (клиент) можно разместить на iPhone, а слои бизнес-логики и базы данных – в центре обработки данных хостинг-провайдера.

6.2.Отображение интерфейса пользователя в веб-браузере

Для реализации слоя бизнес логики в выбранной архитектуре будет использован веб-сервер. Модули веб-сервера (точнее – контейнера сервлетов), реализуемые приложением уточняют класс сервлет, который предназначен для формирования ответов на запросы клиентов. В системе определен специальный вид модулей – серверные страницы, которые формируют HTML страницу с описанием указанных им объектов предметной области. Сформированный HTML передается клиенту для отображения в веб-браузере.

6.3.Отображение упрощенной версии интерфейса пользователя для мобильного браузера

Для доступа к системе из мобильного браузера используется другой адрес URL. Поэтому запросы к системе с обычного веб-браузера обрабатываются одним контроллером, а с мобильного – другим контроллерам, имеющим другой адрес. Контроллер для доступа с мобильного браузера использует отдельный набор страниц для формирования упрощенного HTML для отображения в мобильном браузере.

7.Описание логической структуры и реализация системы

В данном разделе описывается модель системы так, как она будет реализована. Модель показать на диаграмме классов, сгруппировать по всем механизмам обязанности классов и привести их в таблице в разделе 7.1. Пояснить, каким образом обязанности отображаются в операции и атрибуты классов.

В последующих разделах привести заготовки реализации классов. Список классов и операций для реализации согласуется с преподавателем:

- *Реализовать операции для основных сценариев.*

- *Реализовать конструкторы / деструкторы.*

7.1.Логическая структура системы

Привести одну или несколько диаграмм классов для представления различных аспектов системы: структура страниц, классы предметной области и т. д. По сути это одна модель, представленная по частям для удобства.

Привести таблицу с описанием обязанностей классов, собранных из таблиц для кооперaций (разделы 5.x), в которых класс участвует. Пояснить, каким образом объединены разные обязанности.

7.2.Реализация класса Performance

Привести реализации операций класса, список которых согласован с преподавателем. Использовать один из следующих языков:

- *Groovy, JavaScript*
- *Псевдокод*
- *C++, Java, C# - нужно показать, в какие типы языка отображены типы UML.*

Листинг 1. Реализация класса Performance

```
List<Ticket> availableTickets;
List<Ticket> bookedTickets;

private Ticket chooseTicket() {
	if (availableTickets.size() > 0) {
		return availableTickets.get(0);
	}
}

public Ticket bookTicket() {
	Ticket ticket = chooseTicket();

	availableTickets.remove(ticket);
	bookedTickets.add(ticket);
}
```

7.3.Реализация класса Б

Аналогично 7.2

8.Приложение 1. Артефакты проектирования

В приложении нужно привести фотографии / копии артефактов, использованных при разработке, которые подтверждают применение методов проектирования.

Хорошим примером являются карточки CRC, обсуждения на бумаге проектных решений.