Технический Университет Молдовы
Кафедра Автоматики и Информационных Технологий

Курсовой проект
По дисциплине: Анализ и моделирование информационных систем

Тема: «Приложение “Лиманго”»

Выполнил: группа TI-145
Булат Александру

Проверил: ст. преподаватель
Мельник Р.
Сава Н.

Кишинёв 2016

Введение

В курсовом проекте мы собираемся рассмотреть тему приложения которое объединяет в себе функциональность веб-сайта и простоту использования мобильного приложения. В результате работы будут созданы диаграммы такие как диаграмма взаимоотношений, диаграмма вариантов использования, диаграмма кооперации, диаграмма классов, диаграмма активности, диаграмма компонентов, диаграмма состояний и диаграмма развёртывания.

1.Основы UML.

UML появился в результате процесса унификации множества объектно ориентированных языков графического моделирования, процветавших в конце 80-х и в начале 90 х годов. Появившись в 1997 году, он отправил эту Вавилонскую башню в вечность, за что я и многие другие разработчики испытываем по отношению к нему глубокую благодарность.

Способы применения UML.

Основу роли UML в разработке программного обеспечения составляют разнообразные способы использования языка, те различия, которые были перенесены из других языков графического моделирования. Эти отличия вызывают долгие и трудные дискуссии о том, как следует применять UML. Чтобы разрешить эту сложную ситуацию, Стив Меллор (Steve Mellor) и я независимо пришли к определению трех режимов использования

UML разработчиками: режим эскиза, режим проектирования и режим языка программирования. Безусловно, самый главный из трех, по крайней мере, на мой

пристрастный взгляд, – это режим использования UML для эскизирования. В этом режиме разработчики используют UML для обмена информацией о различных аспектах системы. В режиме проектирования можно использовать эскизы при прямой и обратной разработке. При прямой разработке (forward'engineering) диаграммы рисуются до написания кода, а при обратной разработке (re' verse'engineering) диаграммы строятся на основании исходного кода,чтобы лучше понять его. Сущность эскизирования, или эскизного моделирования, в избиратель ности. В процессе прямой разработки вы делаете наброски отдельных элементов программы, которую собираетесь написать, и обычно обсуждаете их с некоторыми разработчиками из вашей команды. При этом с помощью эскизов вы хотите облегчить обмен идеями и вариантами того, что вы собираетесь делать. Вы обсуждаете не всю программу, над которой намереваетесь работать, а только самые важные ее моменты, которые вы хотите донести до коллег в первую очередь, или разделы проекта, которые вы хотите визуализировать до начала программирования. Такие совещания могут быть очень короткими: 10 минутное совещание по нескольким часам программирования или однодневное совещание, посвященное обсуждению двухнедельной итерации. При обратной разработке вы используете эскизы, чтобы объяснить,как работает некоторая часть системы. Вы показываете не все классы, а только те, которые представляют интерес и которые стоит обсудить перед тем, как погрузиться в код. Поскольку эскизирование носит не формальный и динамичный характер и вам нужно делать это быстро и совместно, то наилучшим средством отображения является доска.

Эскизы полезны также и в документации, при этом главную роль играет процесс передачи информации, а не полнота. Инструментами эскизного моделирования служат облегченные средства рисования, и часто разработчики не очень придерживаются всех строгих правил UML. Большинство диаграмм UML, показанных в этой книге, как и в других моих книгах, представляют собой эскизы. Их сила в избирательности передачи информации, а не в полноте описания. Напротив, язык UML как средство проектирования нацелен на полноту. В процессе прямой разработки идея состоит в том, что проект разрабатывается дизайнером, чья работа заключается в построении детальной модели для программиста, который будет выполнять кодирование. Такая модель должна быть достаточно полной в части заложенных проектных решений, а программист должен иметь возможность следовать им прямо и не особо задумываясь. Дизайнером модели может быть тот же самый программист, но, как правило, в качестве дизайнера выступает старший программист, который разрабатывает модели для команды программистов. Причина

такого подхода лежит в аналогии с другими видами инженерной деятельности, когда профессиональные инженеры создают чертежи, которые затем передаются строительным компаниям. Проектирование может быть использовано для всех деталей системы либо дизайнер может нарисовать модель какой то конкретной части. Общий подход состоит в том, чтобы дизайнер разработал модели проектного уровня в виде интерфейсов подсистем, а затем дал возможность разработчикам поработать над реализацией подробностей. При обратной разработке цель моделей состоит в представлении по дробной информации о программе или в виде бумажных документов, или в виде, пригодном для интерактивного просмотра с помощью графического броузера. В такой модели можно показать все детали класса в графическом виде, который разработчикам проще понять. При разработке моделей требуется более сложный инструментарий, чем при составлении эскизов, так как необходимо поддерживать детальность, соответствующую требованиям поставленной задачи. Специализированные CASE средства (computer aided software engineering – автоматизированная разработка программного обеспечения) попадают в эту категорию, хотя сам этот термин стал почти ругательным, и поставщики стараются его избегать. Инструменты прямой разработки поддерживают рисование диаграмм и копирование их в репозиторий с целью со хранения информации. Инструменты обратного проектирования читают исходный код, записывают его интерпретацию в репозиторий и генерируют диаграммы. Инструменты, позволяющие выполнять как прямую, так и обратную разработку, называются двухсторонними(round'trip). Некоторые средства используют исходный код в качестве репозитория, а диаграммы используют его для графического представления. Такие инструменты более тесно связаны с программированием и часто встраиваются прямо в средства редактирования исходного кода. Мне нравится называть их «тройными» инструментами.[1]

Граница между моделями и эскизами довольно размыта, но я думаю, что различия остаются в том, что эскизы сознательно выполняются неполными, подчеркивая важную информацию, в то время как модели нацелены на полноту, часто имея целью свести программирование к простым и до некоторой степени механическим действиям. Короче говоря, я бы определил эскизы как пробные элементы, а модели – как окончательные. Чем дольше вы работаете с UML, а программирование становится все более механическим, тем очевиднее становится необходимость перехода к автоматизированному созданию программ. Действительно многие CASE средства так или иначе генерируют код, что позволяет автоматизировать построение значительной части системы. В конце концов, вы достигнете такой

точки, когда сможете описать с помощью UML всю систему и перейдете в режим использования UML в качестве языка программирования. В такой среде разработчики рисуют диаграммы, которые компилируются прямо в исполняемый код, а UML становится исходным кодом. Очевидно, что такое применение UML требует особенно сложных инструментов. (Кроме того, нотации прямой и об ратной разработки теряют всякий смысл, поскольку UML и исходный код становятся одним и тем же.) Один из интересных вопросов, касающихся UML как языка программирования, – это вопрос о моделировании логики поведения. UML 2 предлагает три способа моделирования поведения: диаграммы взаимодействия, диаграммы состояний и диаграммы деятельности. Все они имеют своих сторонников в сфере программирования. Если UML добьется популярности как язык программирования, то будет интересно посмотреть, какой из этих способов будет иметь успех. Другая точка зрения разработчиков на UML находится где то между его применением для концептуального моделирования и его применением для моделирования программного обеспечения. Большинство разработчиков используют UML для моделирования программного обеспечения. С точки зрения программного обеспечения элементы UML практически непосредственно отображаются в элементы программной системы. Как мы увидим впоследствии, отображение отнюдь не означает следование инструкциям, но когда мы используем UML, мы говорим об элементах программного обеспечения. С концептуальной точки зрения UML представляет описание концепций предметной области. Здесь мы не столько говорим об элементах программного обеспечения, сколько занимаемся созданием словаря для обсуждения конкретной предметной области. Нет строгих правил выбора точки зрения. Поскольку проблему можно рассматривать под разными углами зрения, то и способов применения существует довольно много. Некоторые инструменты автоматически преобразуют исходный код в диаграммы, трактуя UML как альтернативный вид исходного кода.[2]

2. Отношения Use case.

Диаграмма прецедентов (англ. use case diagram, диаграмма вариантов использования) в UML — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.[4]

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.[5]

В данном разделе представлены 4 схемы отношений:

Limango это онлайн магазин разработаный под мобильную платформу Андроид и IOS.

- Это диаграмма в которой лиманго user должен зарегистрироваться , и нажать на кнопку registration если же он зарегистрирован нажмет кнопку Login. А так же пользователь может перейти в Категории и просмотреть какие категории предлагает онлайн магазин. Пользователь может просмотреть каталог продуктов товаров.Гость может выбрать покупку выбрать товар , который хочет купить сам пользователь.

рис 1. Limango user

2)Пользователь может выбрать Registration который содержит в себе два поля ввода первое поле логин input login ввести новый логин это обязательно иначе пользователь не сможет войти в систему.Второе ввод пароля input password это тоже обязательно . После ввода всех полей нажимаем на кнопку Press login и после чего идет переход на ввод логина . Как видим эти поля ввода обязательны include.

рис2. Регистарция пользователя

3)Это Диаграмма содержит в себе ввод логина так же пароля после успешной авторизации мы нажимаем на кнопку Enter и переходим на следующий активити Clothing catalog.

рис 3. Логин

4) После того как мы зарегистрировались мы перешли Список товаров то есть List Products который содержит в себе список разных продуктов товаров Travel содержит список путеовк в горячие туры , Deals Products содержит в себе товары домашних условий то есть мебель и все удобства. Action products который содержит в себе разные вещи что угодно куртки кофты а так же есть выбор для женщин мужчин и детей.Outlet products всяких брендовых товаров.

Items Products он содержит в себе весь список продуктов.

рис 4 . Список товаров

5)В данной диаграмме есть товар сореджит в себе ценv товара price of product содержит картинку товара чтоыб видить какой продукт нам предлагает магазин так же его можно увеличивать чтоыб можно было расмматреть Image of product и скидка на данный продукт скидки в этом онлайн магазине очень часто бывают discount of product .Пользователь может приобрести товар нажав на Choose Buy product и опалат делаться банковским переводом.

рис 5. Товар

6) Категории содержат в себе разные товары женскую одежду women clothes , мужскую одежду Man clothes , мебель furniture, детские вещи Baby's clothes . в Данном магазине есть множественный выбор всего со скидкой .

рис 7. Категории

3. Временные диаграммы.

Временная диаграмма представляет собой симбиоз диаграммы состояний и диаграммы последовательности. Временная диаграмма предназначена для отражения изменения состояний нескольких объектов в ходе взаимодействия во времени.

Существуют два стиля отражения временных диаграмм, которые могут объединяться:

- когда изменения состояния обозначаются переходом от одной горизонтальной линии к другой;
- когда изменения состояния обозначаются перекрещиванием горизонтальных линий.

Первый стиль следует предпочесть, когда состояний немного, а второй стиль лучше подходит, когда количество состояний достаточно велико. Временные ограничения записываются в фигурных скобках. Также на временных диаграммах могут быть указаны сообщения, инициирующие переход объекта из одного состояния в другое.

Симбиозом диаграммы деятельности и диаграммы последовательности стала диаграмма обзора взаимодействия.

В данном разделе представлены 3 схемы отношений:

- В данной диагрмме описываеться как пользователь онлайн магазина

авторизовывается в онлайн магазин и на диаграмме показано как взаимодесвует связь между сервером бразуреом и самим Актером как получает данные отправляет и проверяет.

Рис 1. Авторизация пользователя.

2)В данной диагрмме описывается как пользователь онлайн магазина переход на сам продукт тоесть товар чтоыб увидеть его и посмотреть что он из себя представляет в онлайн магазин и на диаграмме показано как взаимодесвует связь между сервером бразуреом и самим Актером как получает данные отправляет и проверяет.

Рис 2. Переход пользователя на товар.

3) В данной диагрмме описывается как пользователь онлайн магазина ищет в каталоги разные продукты это может быть мебельаня или одежды в онлайн магазин и на диаграмме показано как взаимодесвует связь между сервером бразуреом и самим Актером как получает данные отправляет и проверяет.

Рис 3.Поиск данных по Каталогу

4. Диаграммы кооперации.

Диаграмма коммуникации (англ. communication diagram, в UML 1.x — диаграмма кооперации, collaboration diagram) — диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между объектами, а время как отдельное измерение не используется (применяются порядковые номера вызовов).

В UML есть четыре типа диаграмм взаимодействия:

- Диаграмма последовательности
- Диаграмма коммуникации
- Диаграмма обзора взаимодействия
- Диаграмма синхронизации

Диаграмма коммуникации моделирует взаимодействия между объектами или частями в терминах упорядоченных сообщений. Коммуникационные диаграммы представляют комбинацию информации, взятой из диаграмм классов, последовательности и вариантов использования, описывая сразу и статическую структуру и динамическое поведение системы.

Коммуникационные диаграммы имеют свободный формат упорядочивания объектов и связей как в диаграмме объектов. Чтобы поддерживать порядок сообщений при таком свободном формате, их хронологически нумеруют. Чтение диаграммы коммуникации начинается с сообщения 1.0 и продолжается по направлению пересылки сообщений от объекта к объекту.

Диаграмма коммуникации показывает во многом ту же информацию, что и диаграмма последовательности, но из-за другого способа представления информации какие-то вещи на одной диаграмме видеть проще, чем на другой. Диаграмма коммуникаций нагляднее показывает, с какими элементами взаимодействует каждый элемент, а диаграмма последовательности яснее показывает в каком порядке происходят взаимодействия.[3]

В данной работе представлены 3 схемы кооперации:

1)В данной диграмме описывается как лиманго пользователь переходит по браузеру открывает страничку производиться запрос формы правил приходит формы правил .Ввод логина и пароля отправка данных так же идет проврека данных и приход об успешной авторизации и приход данных лиманго польхователю.

Рис 1. Диагрмма ввода логина и пароля.

2) В данной диграмме описывается как лиманго пользователь переходит по браузеру открывает страничку производиться запрос формы правил приходит формы правил .Ввод данных в посике каталога отправка данных так же идет проврека данных и приход об успешном поиске и приход данных лиманго польхователю.

Рис2. Поиск по каталогу

3) В данной диграмме описывается как лиманго пользователь переходит по браузеру открывает страничку производиться запрос формы правил приходит формы правил .Переход на Item Product чтобы просмотреть данные о товаре отправка данных так же идет проверка данных и приход о получение данных о товаре и приход данных лиманго польхователю.

Рис 3. Данные о товаре

5. Диаграммы Классов.

Диаграмма классов (англ. *Static Structure diagram*) — диаграмма, демонстрирующая классы системы, их атрибуты, методы и взаимосвязи между ними. Входит в UML.

Существует два вида:

- Статический вид диаграммы рассматривает логические взаимосвязи классов между собой;
- Аналитический вид диаграммы рассматривает общий вид и взаимосвязи классов входящих в систему.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

- Концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
- Точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
- Точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).[4]

В данной работе представлены 3 диаграммы Классов:

-

Интерфейс,(операции: оставить коментарий, сделать заказ, читать коментарии)

Рисунок 1.Переход из катогии в класс Clothes

- Данная схема показывает диаграмму возможностей приложения,

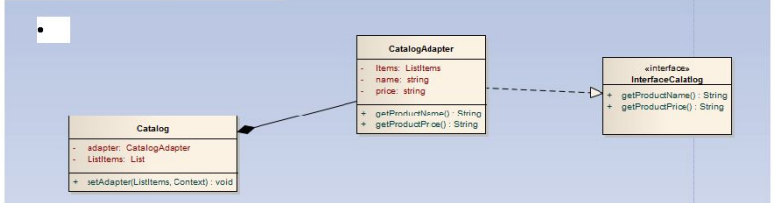
где отношений между классами(Рисунок 11):

Приложение"Лиманго", (атрибуты: операцииб имя продукта , цена продукта)

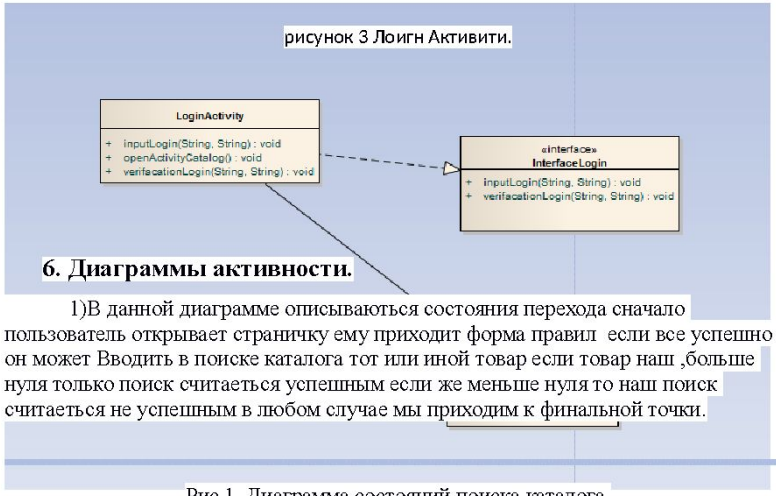
рисунок 2.Каталог

Комментарий, (атрибуты: время сохранения, текст коментария; операции: коментирование, сохранение).

-



Здесь классы ЛогинАктивти в котором происходит верефикация нашего логина инароля и елсивсе хорошо авторизация успешна мы с помощью метода openActivity переходим на другой активти



6. Диаграммы активности.

1)В данной диаграмме описываются состояния перехода сначало пользователь открывает страничку ему приходит форма правил если все успешно он может Вводить в поиске каталога тот или иной товар если товар наш ,больше нуля только поиск считается успешным если же меньше нуля то наш поиск считается не успешным в любом случае мы приходим к финальной точки.

Рис 1. Диаграмма состояний поиска каталога.

2) В данной диаграмме описываются состояния перехода сначало пользователь открывает страничку ему приходит форма правил если все успешно он может Вводить логин и пароль если наш результат ,больше нуля только поиск считается успешным если же меньше нуля то наш поиск считается не успешным в любом случае мы приходим к финальной точки и пользователь авторизован.

Рис 2.Диаграмма состояний Ввода логина и пароля.

3) В данной диаграмме описываються состояния перехода сначало пользователь открывает страничку ему приходит форма правил если все успешно он может перейти на Item Prodcut() чтобы посмотреть данные о товаре если наш результат ,больше нуля только поиск считается успешным если же меньше нуля то наш поиск считается не успешным в любом случае мы приходим к финальной точки.

Рис 3. Диаграмма состояний переход на Item Product.

7.Диаграммы состояний.

1)В данной диаграмме описываються состояния перехода сначало пользователь открывает страничку ему приходит форма правил если все успешно он может Вводить в поиске каталога тот или иной товар если товар наш ,больше нуля только поиск считается успешным если же меньше нуля то наш поиск считается не успешным в любом случае мы приходим к финальной точки.

Рис 1. Диаграмма состояний поиска каталога.

2) В данной диаграмме описываються состояния перехода сначало пользователь открывает страничку ему приходит форма правил если все успешно он может Вводить логин и пароль если наш результат ,больше нуля только поиск считается

успешным если же меньше нуля то наш поиск считается не успешным в любом случае мы приходим к финальной точки и пользователь авторизован.

Рис 2.Диаграмма состояний Ввода логина и пароля.

3) В данной диаграмме описываются состояния перехода сначало пользователь открывает страничку ему приходит форма правил если все успешно он может перейти на Item Prodcut() чтобы посмотреть данные о товаре если наш результат ,больше нуля только поиск считается успешным если же меньше нуля то наш поиск считается не успешным в любом случае мы приходим к финальной точки.

Рис 3. Диаграмма состояний переход на Item Product.

8.Диаграммы Компонентов.

Принципы разработки диаграммы компонентов и диаграммы развёртывания основываются на моделировании архитектуры разрабатываемой системы и физическом представлении программной системы.

Ниже на Рисунке 19 изображена диаграмма компонентов. В данной диаграмме видны взаимоотношения между пользователей и ресторанами реализованные через интерфейс. Рестораны в свою очередь реализовывают классы База отзывов, База графиков и Библиотека Меню.

В данной диаграмме видны взаимоотношения между пользователей и онлайн

магазином реализованные через интерфейс. Магазин в свою очередь реализовывают класс База отзывов

Рисунок 1. Диаграмма Компонентов.

В данной диаграмме видны взаимоотношения между пользователей и онлайн магазином реализованные через интерфейс. Магазин в свою очередь реализовывают класс Базы графиков по продаже товаров

Рисунок 2. Диаграмма Компонентов.

В данной диаграмме видны взаимоотношения между пользователей и онлайн магазином реализованные через интерфейс. Магазин в свою очередь реализовывают класс База графиков

Рисунок 3 Диаграмма Компонентов.

9.Диаграмма Развёртывания.

Ниже на Рисунке 2 изображена диаграмма Развёртывания, которая показывает физическое представление программной системы.

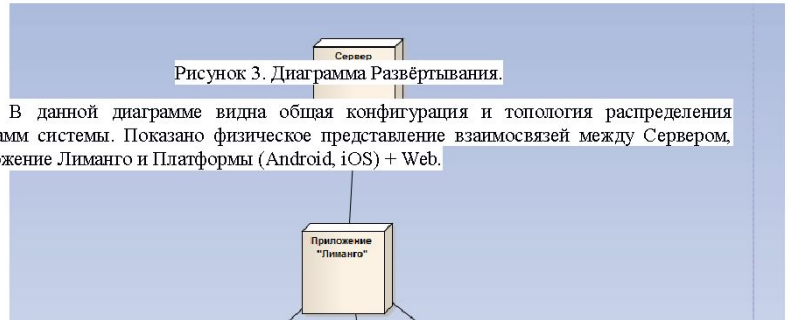


Рисунок 3. Диаграмма Развёртывания.

В данной диаграмме видна общая конфигурация и топология распределения программ системы. Показано физическое представление взаимосвязей между Сервером, Приложение Лиманго и Платформы (Android, iOS) + Web.

Вывод

По окончании исследовательской работы это приложение может быть использовано любой сетью ресторанов. Оно удобное в применении. Все запросы поступают на сервер, что может уменьшить нагрузку на сайт, через который можно осуществить те же функции что и через приложение. Однако стоит заметить, что в наше время большинство людей пользуется гаджетами, а, следовательно, приложение может получить более широкий спрос. Благодаря комментариям можно получить своеобразную рекламу, главное следить за достоверностью информации и своевременно устранять появившиеся недочёты. В данной работе рассмотрели сравнения производственных вопросов и реализация их в приложении. Для улучшения работы системы контроль осуществляется непосредственно беспрерывно. При возникновении любых вопросов моно связаться с администратором приложения (службой поддержки) которая поможет скоординироваться и решить ту или иную проблему.

Мы построили все диаграммы такие как диаграмма взаимоотношений, диаграмма вариантов использования, диаграмма кооперации, диаграмма классов, диаграмма активности, диаграмма компонентов, диаграмма состояний и диаграмма развёртывания.

А так же решили все поставленные задачи.

Список литературы:

- Мартин Фаулер.-UML. Основы, 3-е издание.– Пер. с англ. – СПб: Символ Плюс, 2004. – 192 с.
- www.uml2.ru/faq/uml-основы.html Электронное издание.
- Scott Ambler, Agile Modeling, Wiley, 2002.
- Kent Beck, Extreme Programming Explained: Embrace Change, Addison Wesley, 2000.
- Kent Beck and Martin Fowler, Planning Extreme Programming, Add Addison Wesley, 2002.