Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Специальность: «Программное обеспечение информационных технологий»

**КОНТРОЛЬНАЯ РАБОТА**

По курсу: «Технология разработки программного обеспечения»

На тему: «Мобильное приложения Twishare»

Студент-заочник 2 курса

Группы № 581072

ФИО: Селило Андрей Александрович

Адрес г. Минск,

ул. Радужная д.10, кв. 50

Тел. 8-025-618-26-31

Проверила: Бакунова О.М.

Минск, 2017

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5

1.1 Постановка задачи 5

1.2 Анализ требований к автоматизированной системе 6

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММОГО СРЕДТВА. 7

2.1 Функциональная модель. 9

3 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ 11

3.1 Диаграмма вариантов использования 11

3.2 Диаграмма классов 12

3.3 Диаграмма деятельности 13

3.4 Диаграмма коопераций 14

3.5 Диаграмма последовательности 15

3.6 Диаграмма развёртывания 16

3.7 Диаграмма компонентов 17

4 ГЕНЕРАЦИЯ КОДА 19

4.1 Сгенерированный код 19

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30

**ВВЕДЕНИЕ**

Целью контрольной работы является:

* закрепление знаний, полученных в процессе теоретического изучения дисциплин: «Основы алгоритмизации и программирования», «Конструирование программ и языки программирования», «Технология разработки ПО»;
* знакомство с современными средствами автоматизации процесса проектирования и разработки ПО;
* закрепление изученных методов и приёмов в области проектирования ПО;
* закрепление знаний международных и отечественных стандартов;
* решение задач на этапе сопровождения программного продукта;
* изучение предметной области по теме;
* разработка UML диаграмм.

UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

База данных на сегодняшний день – это самый распространенный «повод» для написания программ. Все современные языки программирования содержат в себе встроенные возможности для быстрого и удобного создания СУБД.

СУБД – это, конечно, нечто большее, чем просто набор структурированных данных. СУБД можно назвать «умной» средой, с помощью которой можно обрабатывать, искать, передавать, хранить данные. От качества СУБД зависит эффективность работы с базой данных в целом. Современные СУБД – это «высокоинтеллектуальные» системы, позволяющие работать не только с базами данных, но и с базами знаний. Это направление, связанное с накоплением, получением, сортировкой и использованием знаний, является довольно новым. С понятием «база знаний» тесно связанно понятие искусственного интеллекта, что говорит об огромных масштабах работ, которые потребуются провести для совершенствования сегодняшних СУБД, так как искусственный интеллект – это новый горизонт в нашей компьютерной эпохе. При работе баз данных и связанных с ними программ обработки и поиска информации необходимо учитывать специфику предметной области – одно из важнейших достоинств базы, данных. Конечно, здесь можно говорить о том, что уже давно существуют универсальные формы для баз данных, универсальные СУБД, универсальный модуль обработки данных. Но эта универсальность требует больших, (если не огромных) затрат машинного времени и ресурсов, не говоря уж о стоимости всего ПО. Поэтому чаще всего используют специализированные БД и СУБД.

Rational Rose представляет собой CASE средство проектирования и разработки информационных систем и программного обеспечения для управления предприятиями. Как и другие CASE средства (ARIS, BPwin, ERwin) его можно применять для анализа и моделирования бизнес процессов. Первая версия этого продукта была выпущена компанией Rational Software . В дальнейшем Rational Rose был куплен IBM.

Принципиальное отличие Rational Rose от других средств заключается в объектно-ориентированном подходе. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке UML (Unified Modeling Language). Инструменты моделирования Rational Rose позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии.

В данной контрольной работе также будет создана специализированная база данных для мобильного приложения Twishare.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Требуется разработать мобильное приложение, позволяющее создавать шаблонные сообщения и оправлять их через социальную сеть Twitter. Для реализации цели необходимо выполнить ряд задач:

1. Разработать простой и понятный интерфейс;

2. Проработать логику работы приложения в нестандартных условиях;

3. Протестировать и отладить приложение;

4. Составить сопроводительную документацию.

Мобильное приложение «Twishare» в конечном результате должно обладать следующими функциональными характеристиками:

* Простой, удобный и функциональный интерфейс с подсказками, который позволит пользователям без лишних проблем разобраться в функциях приложения;
* Возможность добавления шаблона;
* Возможность отправки сообщения в одно касание;
* Оповещения о совершенных действиях пользователя посредствам уведомлений;
* Смена аккаунта пользователя в настройках приложения;
* Отображение на «плитке» для отправки сообщения текста, который назначен на нее;

В качестве языка программирования был выбран C#, как более адаптированный для реализации данной задачи. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

## **Постановка задачи**

В данной контрольной работе необходимо реализовать базу данных программного средства «Twishare».

* 1. **Анализ требований к автоматизированной системе**

Всю работу мобильного приложения «Twishare» можно разделить на несколько основных подпунктов:

* Авторизация в приложении;
* Написание текста шаблона, или выбор из списка готовых шаблонов;
* Расположение «плитки» на рабочем экране смартфона;
* Отправка уведомления при помощи «тапа» по соответствующей плитке на рабочем экране смартфона.

В основу конструирования мобильного приложения «Twishare» положены следующие основные принципы:

* Отправка оповещений в одно касание;
* Пред. просмотр отправляемого оповещения;
* Включение в оповещение местоположения пользователя;
* Возможность написания пользовательских оповещений, а также присутствие шаблонов с уже готовыми оповещениями;
* Смена пользователя.

Процесс проектирования архитектуры программного обеспечения состоит в проектировании структуры всех его компонент, функционально связанных с решаемой задачей, включая сопряжения между ними и требования к ним.

Архитектура программного обеспечения в традиционном смысле включает определение всех модулей программ, их иерархии и сопряжения между ними и данными.

Во время разработки архитектуры программного обеспечения выполняется его модульно-иерархическое построение.

1. **СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

В работе проводится моделирование с использование IDEF0(BPWin), UML (Rational Rose 2000), IDEF1x (ErWin).

1. Важная роль отводится процессу функционального проектирования.

Для регламентирования создания функциональных моделей ПС предназначен стандарт IDEF0 (Integrated Definition Function Modeling), который и реализован в пакете BpWin.

В основе IDEF0 лежит понятие блока, который реализует некую конкретную функцию. Четыре стороны блока имеют разное назначение. Слева отображаются входные данные (исходные данные). Справа – выходные данные (результат выполнения функции). Сверху – управление (управляющие воздействия на функцию). Снизу – механизм (посредством чего реализуется данная функция).

Функция – это управляемое действие над входными данными, результатом которого являются выходные данные, при этом используется некий механизм. Взаимодействие между функциями отображается в виде стрелок. Иногда стороны блока называют направлениями, а стрелки потоками. Стрелки можно подписывать. Подписи связываются с конкретной стрелкой при помощи зигзага.

В основе IDEF0 лежит три базовых принципа:

1) принцип функциональной декомпозиции – любая функция может быть разбита (декомпозирована) на более простые функции (более понятен термин детализация);

2) принцип ограничения сложности – количество блоков на диаграмме должно быть не менее двух, но не более шести (условие удобочитаемости);

3) принцип контекста – моделирование делового процесса начинается с построения контекстной диаграммы, на которой отображается только один блок – главная функция моделирующей системы, ограничивающая область границы моделирующей системы (регламентирует начальный этап построения модели).

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что является внешним воздействием на систему. Т. е. первоначально нужно определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели.

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой мы называем универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друх от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира. Примером сущности IDEF1X может быть сущность "СОТРУДНИК", которая представляет собой всех сотрудников предприятия, а один из них, скажем, Иванов Петр Сергеевич, является конкретной реализацией этой сущности.

Сase-средство BPwin значительно облегчает задачу создания информационной системы, позволяя осуществить декомпозицию сложной системы на более простые с тем, чтобы каждая из них могла проектироваться независимо, и для понимания любого уровня проектирования достаточно было оперировать с информацией о немногих ее частях.

Рассмотрим процесс моделирования в методологии IDEF0 на примере контекстной диаграммы разработанной модели процесса учета кредитных операций в банке.

Цель модели – моделирование процесса учета выданных кредитов.

Точка зрения модели – банковский служащий, отвечающий за выдачу кредитов.

Входы модели (диаграммы):

1. Паспортные данные получателя кредита – содержит информацию о получателе кредита;
2. Кредитная история – сведения о лицах, которые уже брали ранее у банка кредита для определения степени благонадежности получателя кредита;
3. Пожелания получателя кредита – определяют цель кредитования, размер кредита, сроки погашения и т.п.;
4. Время – все этапы процесса учета требуют затрат времени.

Процесс учета регламентируется порядком выдачи кредита и порядком учета кредитных операций в банке.

Выходные процессы учета – запись об учтенной кредитной операции, информация об ошибке и отчетная документация.

* 1. **Функциональная модель**

Проанализировав как должна работать система в целом, была построена модель A0 на рисунке 2.1.

В качестве входных данных представлены:

* нажатие пользователя на «плитку»;
* ввод пользователем логина и пароля;
* ввод или выбор оповещения.

В качестве выходных данных представлены:

* отправка сообщения.

В качестве механизмов представлены:

* пользователь.

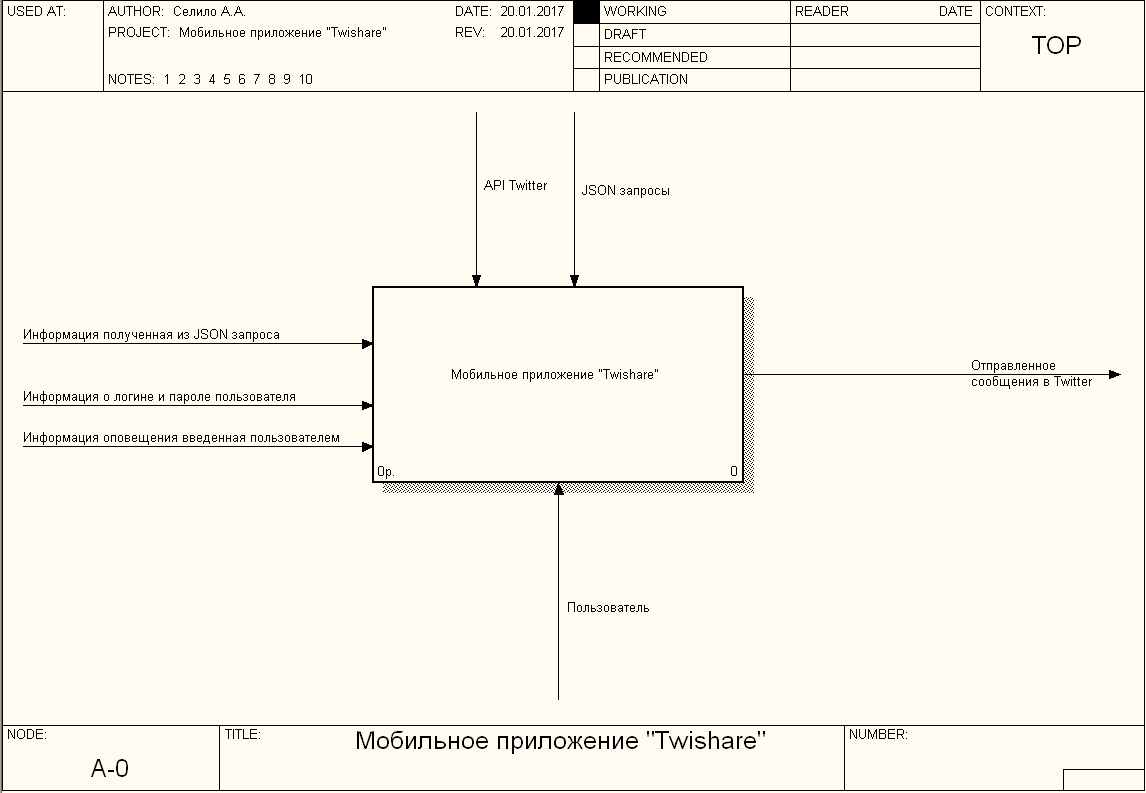
****

Рисунок 2.1 - Диаграмма модели A-0

Декомпозиция модели A0 изображена на рисунке 2.2 , где отображены основные подпроцессы основного бизнес-процесса.

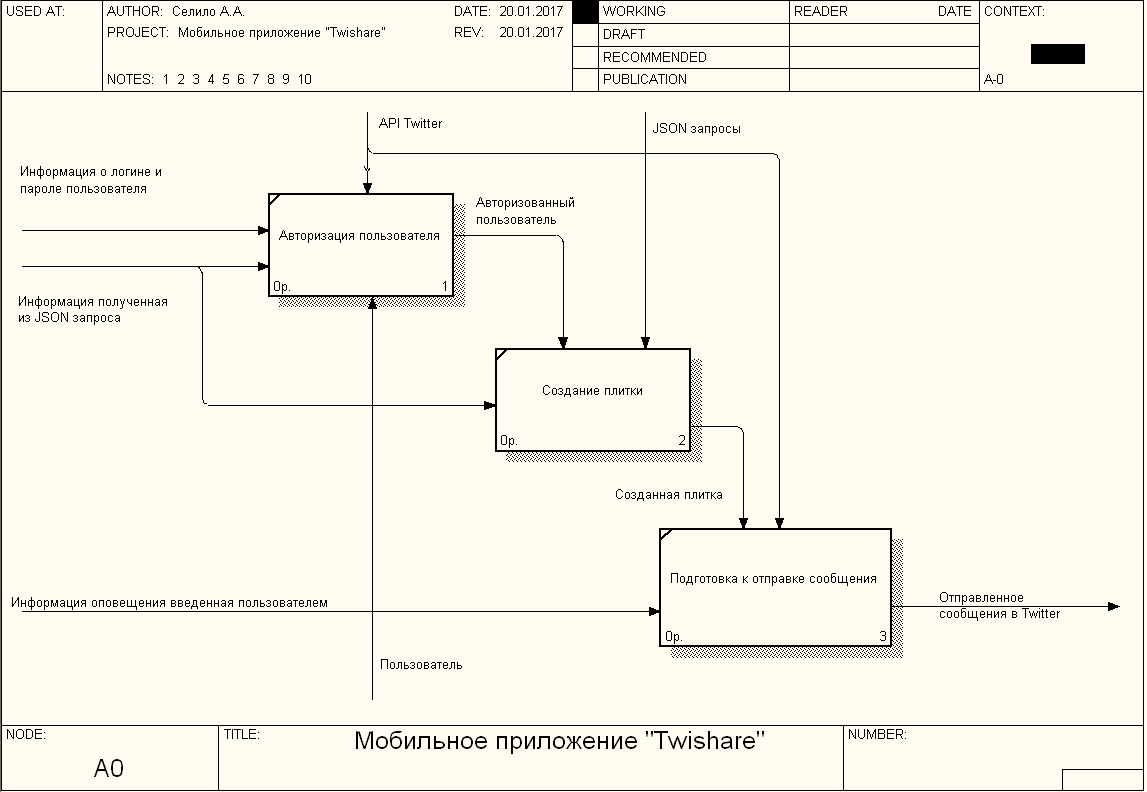
****

Рисунок 2.2 - Декомпозиция модели A0

1. **ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ**

### **3.1 Диаграмма вариантов использования**

Модель вариантов использования предназначается для определения требований к системе. Для отображения этой модели язык UML предлагает использовать диаграммы Use Case (вариант использования) совместно с моделями State Diagram (диаграммы состояний) и Activity Diagram (диаграммы деятльности/активности).

Разработка диаграммы вариантов использования преследует цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования представлена на рисунке 3.1

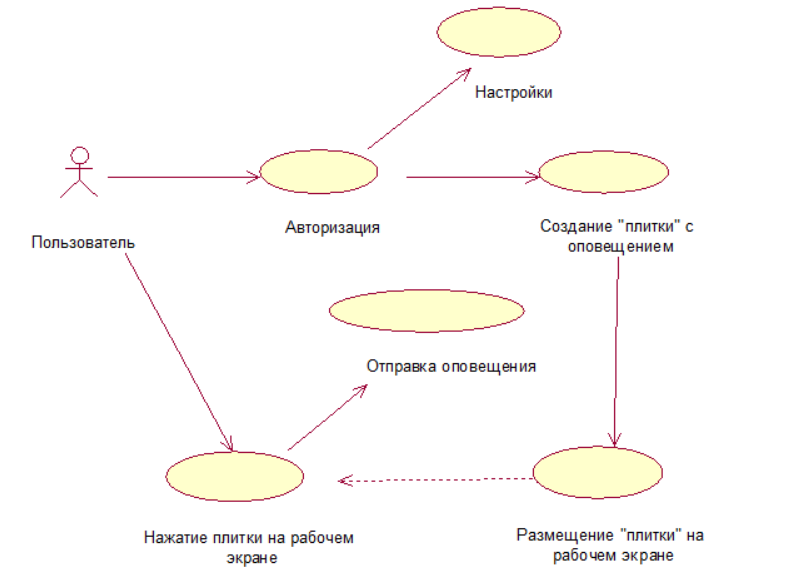


Рисунок 3.1 – Диаграмма вариантов использования

### **3.2 Диаграмма классов**

Диаграмма классов — 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).

Диаграмма классов представлена на рисунке 3.2

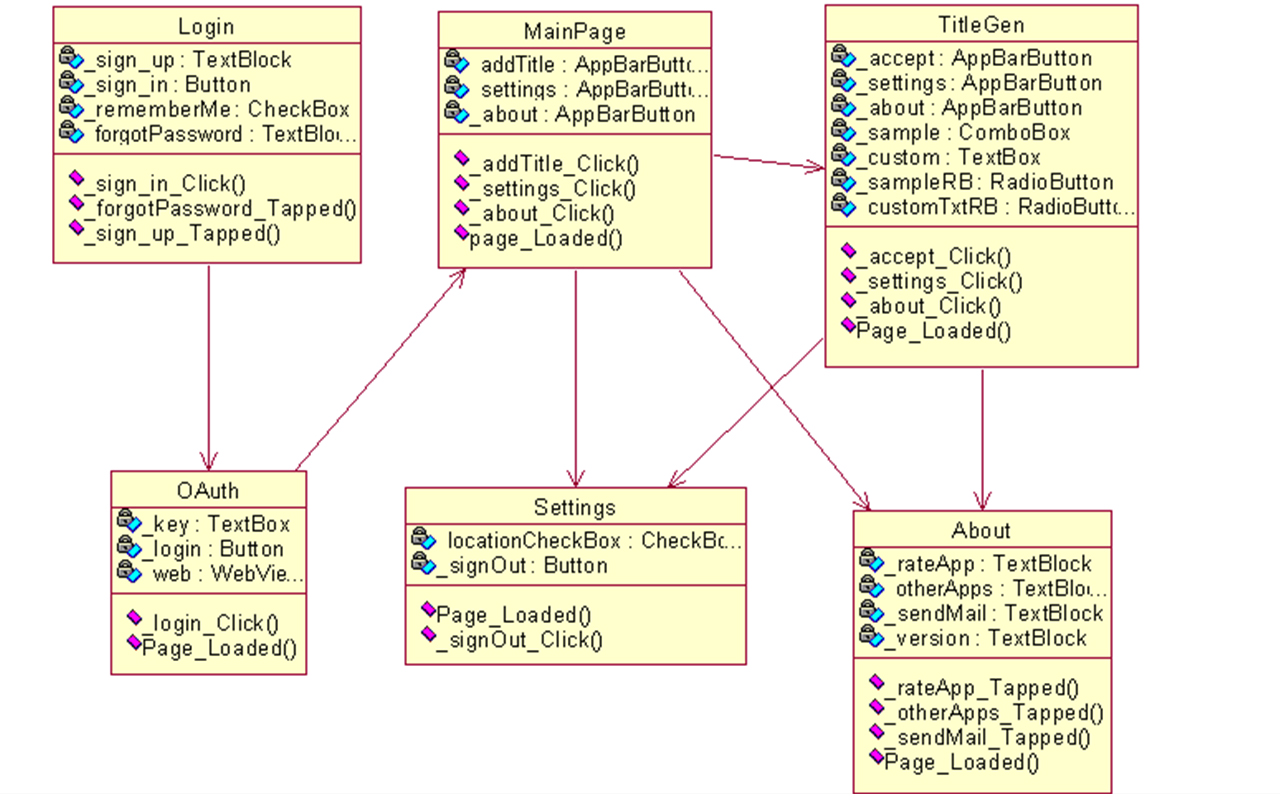


Рисунок 3.2 – Диаграмма классов

### **3.3 Диаграмма деятельности**

Диаграмма деятельности — UML-диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (англ. activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграмма деятельности представлена на рисунке 3.3

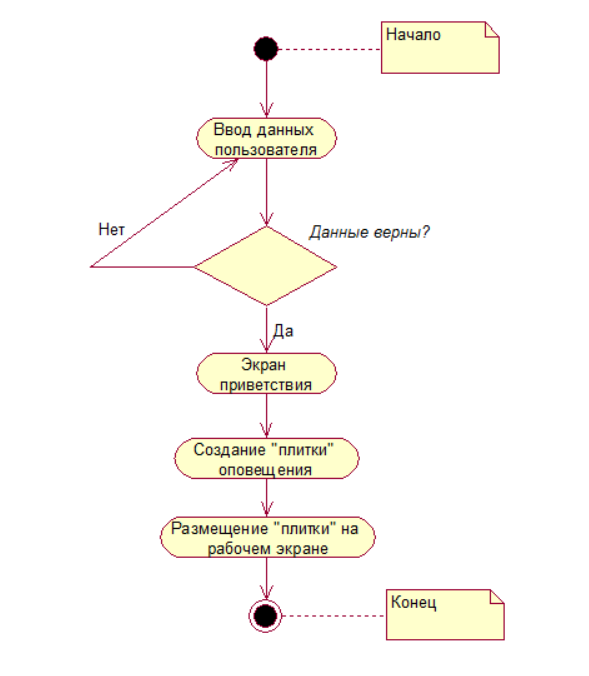
****

Рисунок 3.3 – Диаграмма деятельности

### **3.4 Диаграмма коопераций**

Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме кооперации в виде прямоугольников изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи - потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. На этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров. Следовательно, если необходимо явно специфицировать взаимосвязи между объектами в реальном времени, лучше это делать на диаграмме последовательности.

Диаграмма коопераций представлена на рисунке 3.4

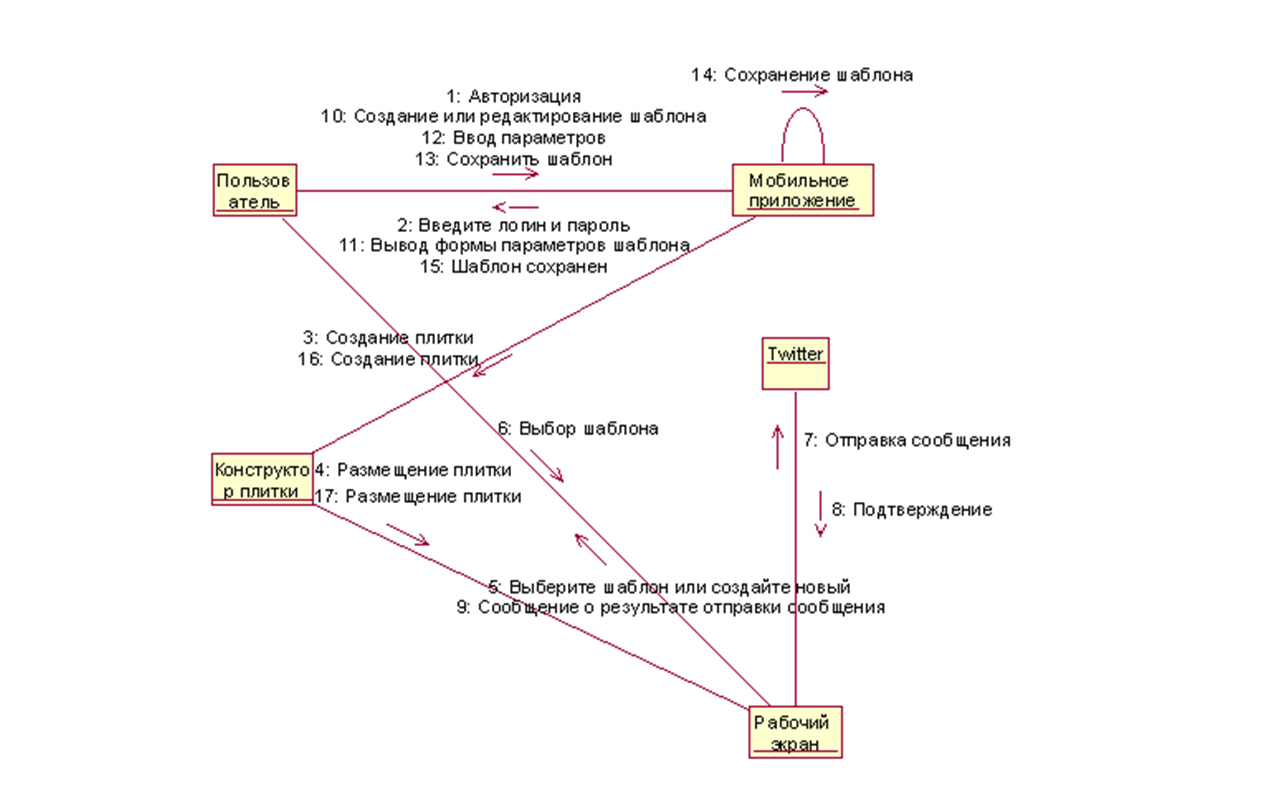


Рисунок 3.4 – Диаграмма коопераций

### **3.5 Диаграмма последовательности**

Диаграмма последовательности — диаграмма, на которой показано взаимодействие объектов, упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления. Используется в языке UML.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Диаграмма последовательности представлена на рисунке 3.5

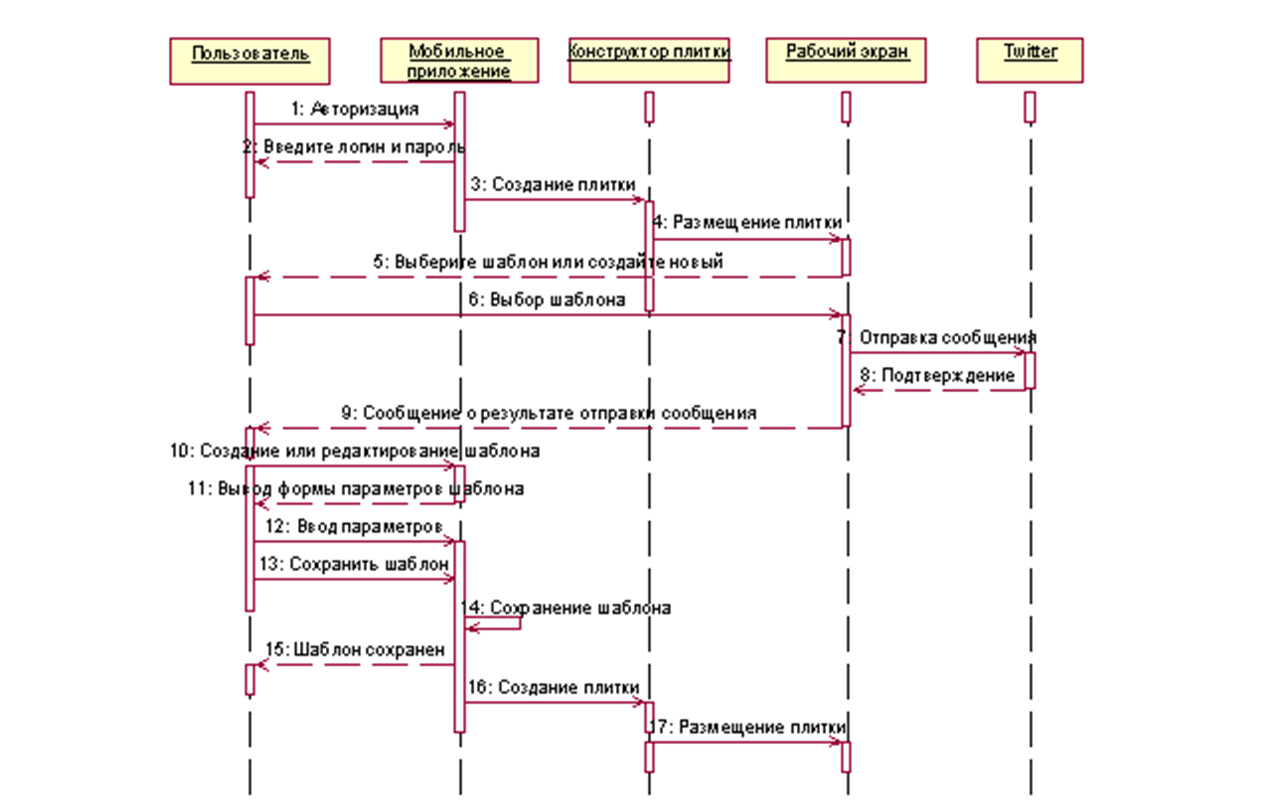


Рисунок 3.5 – Диаграмма последовательности

### **Диаграмма развёртывания**

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь под узлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

* узел устройства;
* узел среды выполнения.

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Диаграмма развёртывания представлена на рисунке 3.6

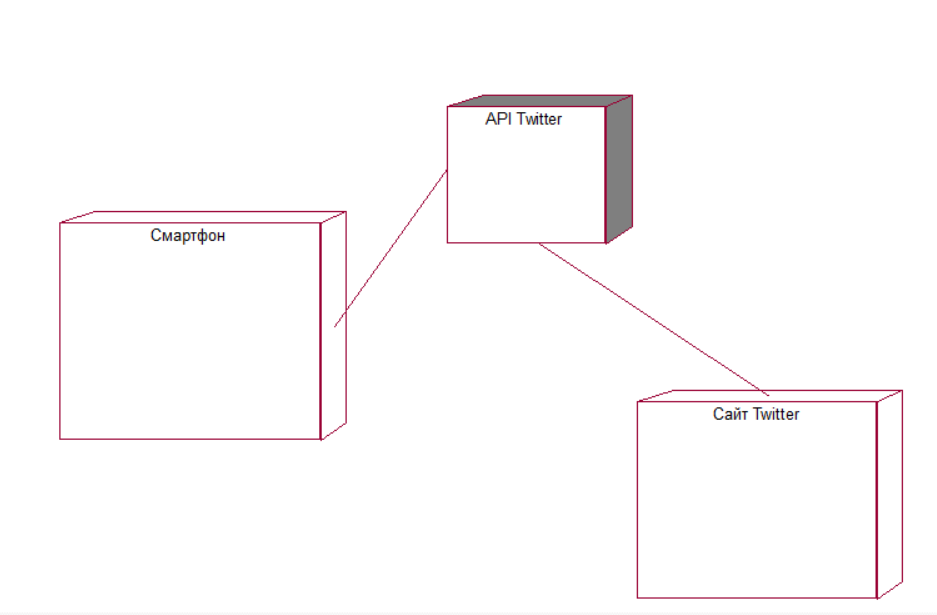
****

Рисунок 3.6 – Диаграмма развёртывания

### **Диаграмма компонентов**

Диаграмма компонентов — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывается связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами.

Диаграмма компонентов представлена на рисунке 3.7.1.

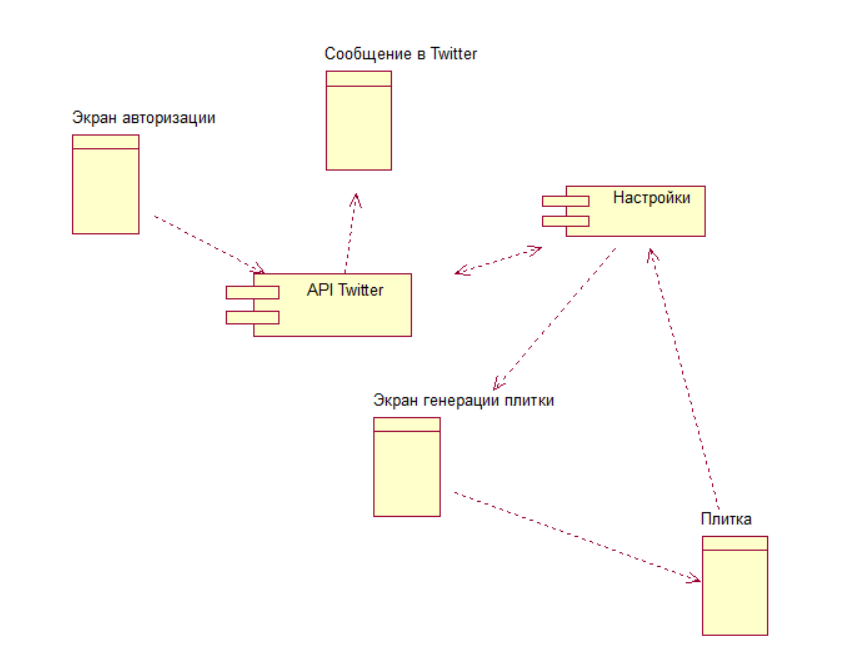
****

Рисунок 3.7 – Диаграмма компонентов

### **ГЕНЕРАЦИЯ КОДА**

На рисунках 4.1 и 4.2 представлен процесс генерации кода на языке С++, непосредственно сам код расположен ниже.

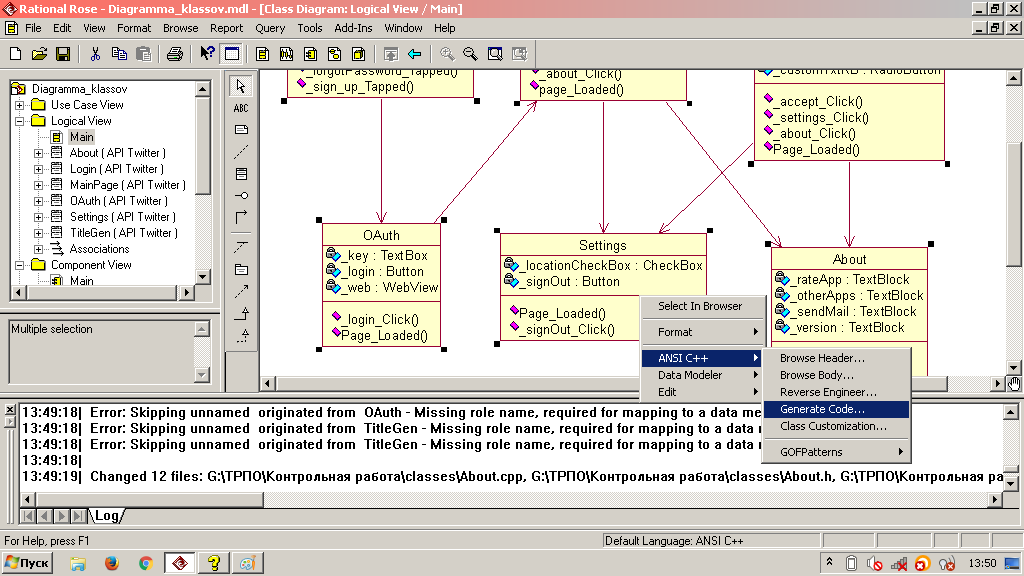


Рисунок 4.1 – Процесс генерации кода

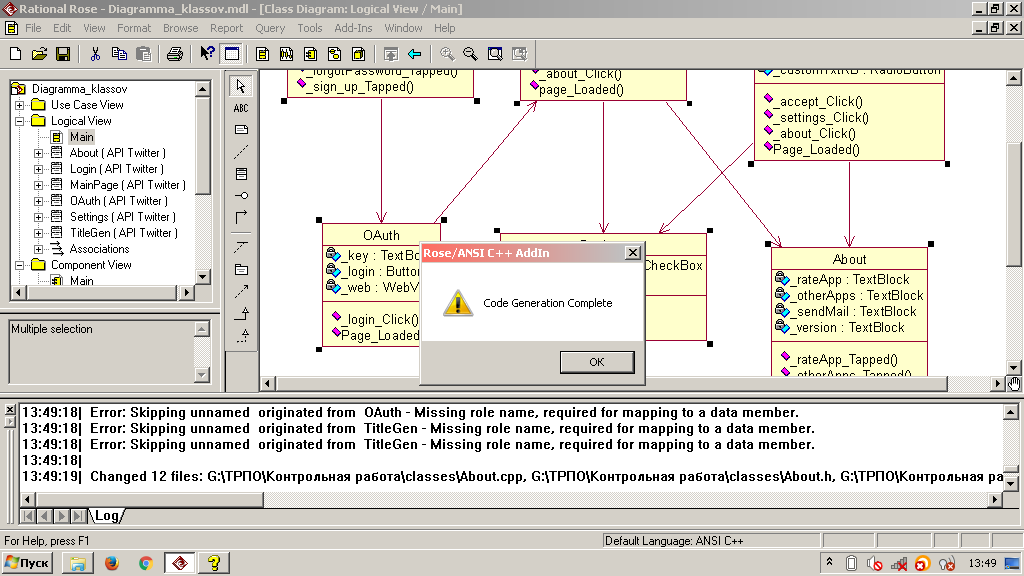


Рисунок 4.2 – Процесс генерации кода

**4.1 Сгенерированный код**

Файл About.h:

#ifndef ABOUT\_H\_HEADER\_INCLUDED\_A785B70F

#define ABOUT\_H\_HEADER\_INCLUDED\_A785B70F

//##ModelId=57EF8D10033D

class About

{

public:

//##ModelId=5879CCB300B3

void \_rateApp\_Tapped();

//##ModelId=5879CCCE0068

void \_otherApps\_Tapped();

//##ModelId=5879CCCE03E1

void \_sendMail\_Tapped();

//##ModelId=5879CCD00055

void Page\_Loaded();

private:

//##ModelId=57EF8D1D002C

TextBlock \_rateApp;

//##ModelId=5879CC650390

TextBlock \_otherApps;

//##ModelId=5879CC7900A2

TextBlock \_sendMail;

//##ModelId=5879CC840207

TextBlock \_version;

};

#endif /\* ABOUT\_H\_HEADER\_INCLUDED\_A785B70F \*/

Файл About.cpp:

#include "About.h"

//##ModelId=5879CCB300B3

void About::\_rateApp\_Tapped()

{

}

//##ModelId=5879CCCE0068

void About::\_otherApps\_Tapped()

{

}

//##ModelId=5879CCCE03E1

void About::\_sendMail\_Tapped()

{

}

//##ModelId=5879CCD00055

void About::Page\_Loaded()

{

}

Файл Login.h:

#ifndef LOGIN\_H\_HEADER\_INCLUDED\_A78596A9

#define LOGIN\_H\_HEADER\_INCLUDED\_A78596A9

//##ModelId=57EF8E320260

class Login

{

public:

//##ModelId=5879C5060079

void \_sign\_in\_Click();

//##ModelId=5879C54E0073

void \_forgotPassword\_Tapped();

//##ModelId=5879C570000E

void \_sign\_up\_Tapped();

private:

//##ModelId=57EF8E3E0269

TextBlock \_sign\_up;

//##ModelId=57EF8E4703A0

Button \_sign\_in;

//##ModelId=57EF8E4E00DC

CheckBox \_rememberMe;

//##ModelId=57EF8E550035

TextBlock \_forgotPassword;

};

#endif /\* LOGIN\_H\_HEADER\_INCLUDED\_A78596A9 \*/

Файл Login.cpp:

#include "Login.h"

//##ModelId=5879C5060079

void Login::\_sign\_in\_Click()

{

}

//##ModelId=5879C54E0073

void Login::\_forgotPassword\_Tapped()

{

}

//##ModelId=5879C570000E

void Login::\_sign\_up\_Tapped()

{

}

Файл MainPage.h:

#ifndef MAINPAGE\_H\_HEADER\_INCLUDED\_A785D8A6

#define MAINPAGE\_H\_HEADER\_INCLUDED\_A785D8A6

//##ModelId=57EF8E1401C7

class MainPage

{

public:

//##ModelId=5879C68E0048

void \_addTitle\_Click();

//##ModelId=5879C6E90310

void \_settings\_Click();

//##ModelId=5879C6F7022F

void \_about\_Click();

//##ModelId=5879C7320125

void page\_Loaded();

private:

//##ModelId=57EF8E1C02DA

AppBarButton \_addTitle;

//##ModelId=57EF8E2102EE

AppBarButton \_settings;

//##ModelId=57EF8E27007E

AppBarButton \_about;

};

#endif /\* MAINPAGE\_H\_HEADER\_INCLUDED\_A785D8A6 \*/

Файл MainPage.cpp:

#include "MainPage.h"

//##ModelId=5879C68E0048

void MainPage::\_addTitle\_Click()

{

}

//##ModelId=5879C6E90310

void MainPage::\_settings\_Click()

{

}

//##ModelId=5879C6F7022F

void MainPage::\_about\_Click()

{

}

//##ModelId=5879C7320125

void MainPage::page\_Loaded()

{

}

Файл OAuth.h:

#ifndef OAUTH\_H\_HEADER\_INCLUDED\_A785D3B2

#define OAUTH\_H\_HEADER\_INCLUDED\_A785D3B2

//##ModelId=57EF89B10210

class OAuth

{

public:

//##ModelId=5879CA2D0289

void \_login\_Click();

//##ModelId=5879CA49005A

void Page\_Loaded();

private:

//##ModelId=57EF8C3201FB

TextBox \_key;

//##ModelId=57EF8C390245

Button \_login;

//##ModelId=57EF8C46012C

WebView \_web;

};

#endif /\* OAUTH\_H\_HEADER\_INCLUDED\_A785D3B2 \*/

Файл OAuth.cpp:

#include "OAuth.h"

//##ModelId=5879CA2D0289

void OAuth::\_login\_Click()

{

}

//##ModelId=5879CA49005A

void OAuth::Page\_Loaded()

{

}

Файл Settings.h:

#ifndef SETTINGS\_H\_HEADER\_INCLUDED\_A7858C89

#define SETTINGS\_H\_HEADER\_INCLUDED\_A7858C89

//##ModelId=57EF8C6B022C

class Settings

{

public:

//##ModelId=5879CBD301F9

void Page\_Loaded();

//##ModelId=5879CBF50185

void \_signOut\_Click();

private:

//##ModelId=57EF8C7900FE

CheckBox \_locationCheckBox;

//##ModelId=57EF8C7E0334

Button \_signOut;

};

#endif /\* SETTINGS\_H\_HEADER\_INCLUDED\_A7858C89 \*/

Файл Settings.cpp:

#include "Settings.h"

//##ModelId=5879CBD301F9

void Settings::Page\_Loaded()

{

}

//##ModelId=5879CBF50185

void Settings::\_signOut\_Click()

{

}

Файл TitleGen.h:

#ifndef TITLEGEN\_H\_HEADER\_INCLUDED\_A785CDCF

#define TITLEGEN\_H\_HEADER\_INCLUDED\_A785CDCF

//##ModelId=57EF8DD50309

class TitleGen

{

public:

//##ModelId=5879C8B503BA

void \_accept\_Click();

//##ModelId=5879C8EA02E8

void \_settings\_Click();

//##ModelId=5879C8ED0006

void \_about\_Click();

//##ModelId=5879C8ED019C

void Page\_Loaded();

private:

//##ModelId=57EF8DE2035E

AppBarButton \_accept;

//##ModelId=5879C7D4030F

AppBarButton \_settings;

//##ModelId=5879C7EB03C5

AppBarButton \_about;

//##ModelId=5879C7FC032D

ComboBox \_sample;

//##ModelId=5879C8130124

TextBox \_custom;

//##ModelId=5879C85D00FC

RadioButton \_sampleRB;

//##ModelId=5879C873031A

RadioButton \_customTxtRB;

};

#endif /\* TITLEGEN\_H\_HEADER\_INCLUDED\_A785CDCF \*/

Файл TitleGen.cpp:

#include "TitleGen.h"

//##ModelId=5879C8B503BA

void TitleGen::\_accept\_Click()

{

}

//##ModelId=5879C8EA02E8

void TitleGen::\_settings\_Click()

{

}

//##ModelId=5879C8ED0006

void TitleGen::\_about\_Click()

{

}

//##ModelId=5879C8ED019C

void TitleGen::Page\_Loaded()

{

}

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. ГОСТ 2.105-95 ЕСКД. Введ. 1996–07–01. – М.: Изд-во стандартов, 1996.
2. ГОСТ 19.104–78 – Единая система программной документации. Основные надписи.
3. Маклаков С.В. BPwin и ERwin: CASE - средства для разработки информационных систем.
4. Федотов Д.Э., Семенов Ю.Д., Чижик К.Н. Практикум для высших учебных заведений. CASE-технологии.-157с.
5. Интернет обучение [Электронный ресурс] / Котляр Д.С. Использование CASE-средства ERwin для автоматизации проектирования и разработки базы данных – Режим доступа: http://royallib.com/read/ bezopasnost – Дата доступа: 15.12.2016.
6. Интернет обучение [Электронный ресурс] / Информационные системы и технологии – Режим доступа: <http://www.narfu.ru> – Дата доступа: 19.12.2016.
7. Интернет обучение [Электронный ресурс] / Нормализация структурны данных – Режим доступа: http://infostart.ru/public/269803/ – Дата доступа: 15.01.2016.
8. Интернет обучение [Электронный ресурс] / Программа компьютерного моделирования BpWin – Режим доступа: http://bourabai.kz/cm/bpwin.htm– Дата доступа: 11.11.2016.
9. Леоненков. «Самоучитель UML».
10. 2. Б. Хичков SYBASE: Настольная книга администратора. Лори, 2000 г. 448 с.
11. 3. Р. Мюллер. Базы данных и UML: Проектирование.– Лори, 2002г. 432 с.
12. 4. Фельдман С.К. Система программирования Java без секретов: Как создать безопасное приложение с "нуля". – Новый издательский дом" , 2005 г. , 347 с.
13. 5. Дейтел П.Дж., Дейтел Х.М. Как програмировать на Java. Книга 2. Файлы, сети, базы данных. – "Бином" · 2005 г., 672 с.
14. Интернет обучение [Электронный ресурс] / Основные методологии обследования организаций. Стандарт IDEF0. – Режим доступа: http://www.cfin.ru/vernikov/idef/idef0.shtml
15. Интернет обучение [Электронный ресурс] / Основы методологии IDEF1X. – Режим доступа: http://www.cfin.ru/vernikov/idef/idef0.shtml