Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Программное средство «Twitter - клиент»

БГУИР КР 1-40 01 01 523 ПЗ

Студент: гр. 651005 Стаселович И. А.

Руководитель: Балашко А. О.

Минск 2018

**ВВЕДЕНИЕ**

На сегодняшний день одним из неотъемлемых частей нашей повседневной жизни стал персональный компьютер. Его можно использовать для редактирования фото и просмотра видео, набора документов и чтения книг, серфинга в сети Интернет и игры в видеоигры.

В наше время, когда каждая минута драгоценна, такая техника существенно расширяет возможности человека. Персональный компьютер с доступом к сети Интернет даёт возможность доступа к любой информации. С его помощью можно перевести или получить денежные средства, найти нужного человека с другого конца света, никогда не терять связь с семьей и знакомыми, поговорить по видеосвязи и устроить видеоконференцию в офисе.

С популяризацией ПК, его использование для досуга и развлечения стало нормой в современном обществе. Пользователь может насладиться фильмом или прослушать новый альбом любимого исполнителя, не выходя из дома.

С помощью ПК можно повысить эффективность многих видов работы, чем пользуется большое количество людей. Наличие ПК – это всегда удобно и многие в наше время не могут представить свою жизнь без него.

С высоким распространением социальных сетей появилась потребность постоянно иметь доступ к своему аккаунту и оставаться в курсе последних событий. При этом удобно иметь под рукой клиент, который обладает простым функционалом и способен выполнять основные функции, которые присущи социальной сети. Таким образом, был разработан Твиттер - клиент, способный просматривать ленту новостей, посылать сообщения, а также обладающий некоторыми другими функциями социальной сети Twitter.

**1 АНАЛИТИЧЕСКАЯ ЧАСТЬ**

* 1. **Обзор существующих Твиттер – клиентов**

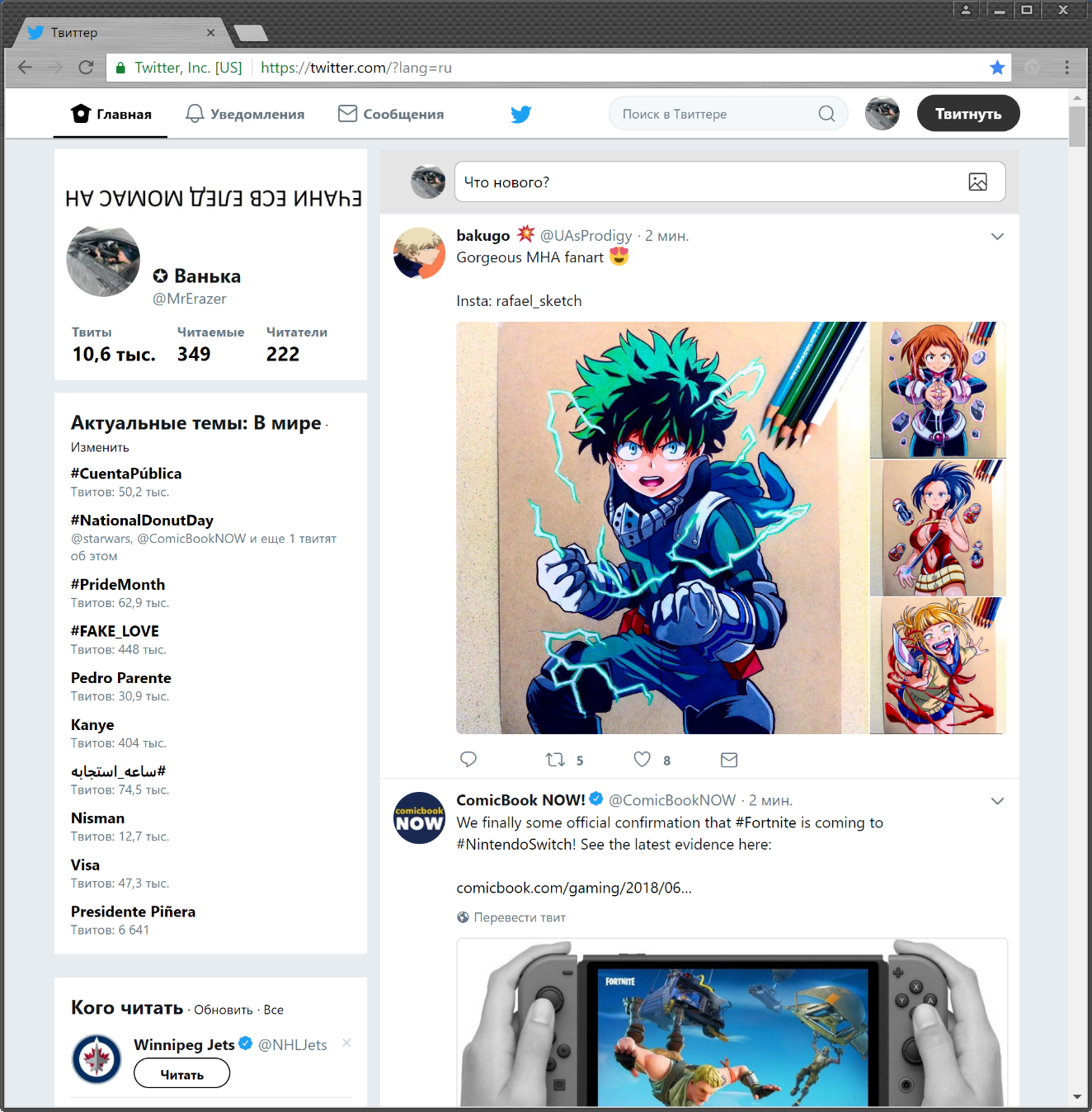
Невзирая на то, что официальный сайт Твиттера (рис. 1.1), предоставляет интерфейс, достаточный для полноценного взаимодействия пользователя с ним, существуют и альтернативные оболочки – такие, как расширения, сервисы и программы для разных операционных систем. Большую часть Твиттер - клиентов занимают приложения для мобильных устройств на операционных системах Android и IOS.

Рисунок 1.1 – официальный сайт Твиттер

Одним из наиболее известных и любимых аудиторией альтернативных Твиттер-клиентов для пользователей разных платформ является приложение Tweetbot (рис 1.2).



Рисунок 1.2 – приложение Tweetbot для MacOS

Этот полнофункциональный клиент имеет отличную поддержку нескольких учетных записей и списков. Он также имеет мощные фильтры, несколько видов столбцов, и многое другое. Также имеется версия для устройств на операционной системой IOS (рис 1.3).

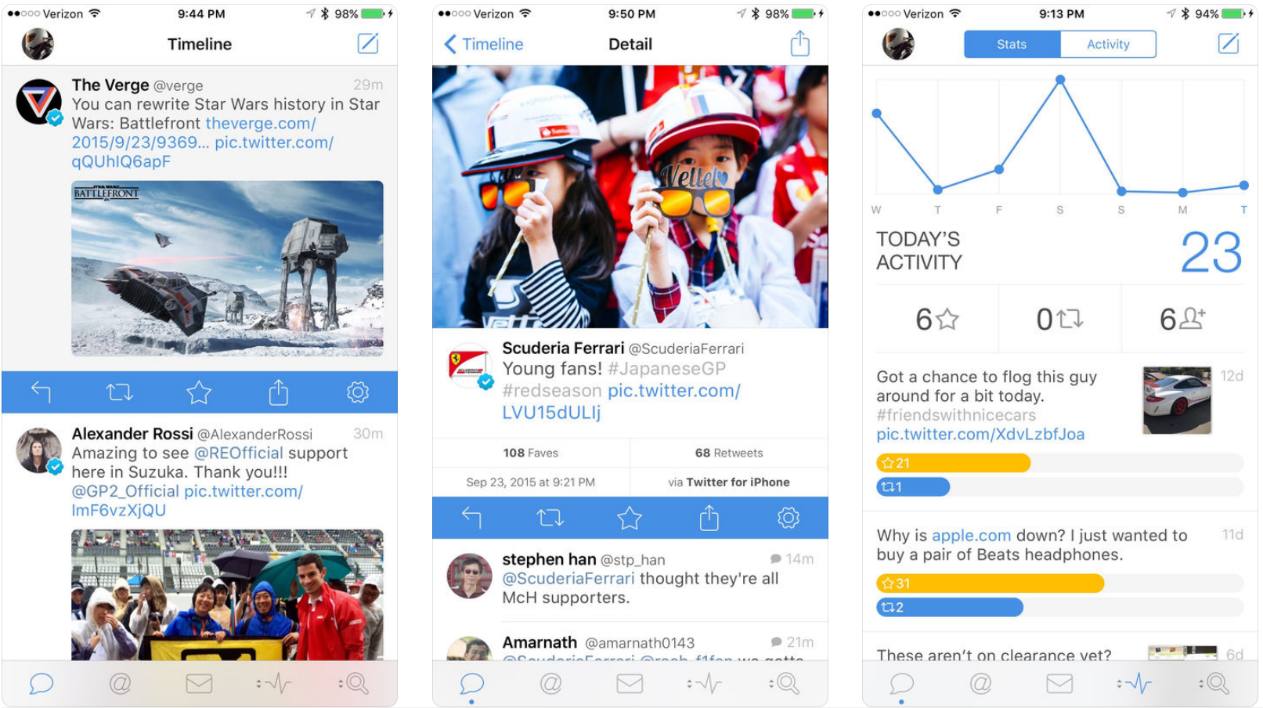


Рисунок 1.3 – приложение Tweetbot для IOS

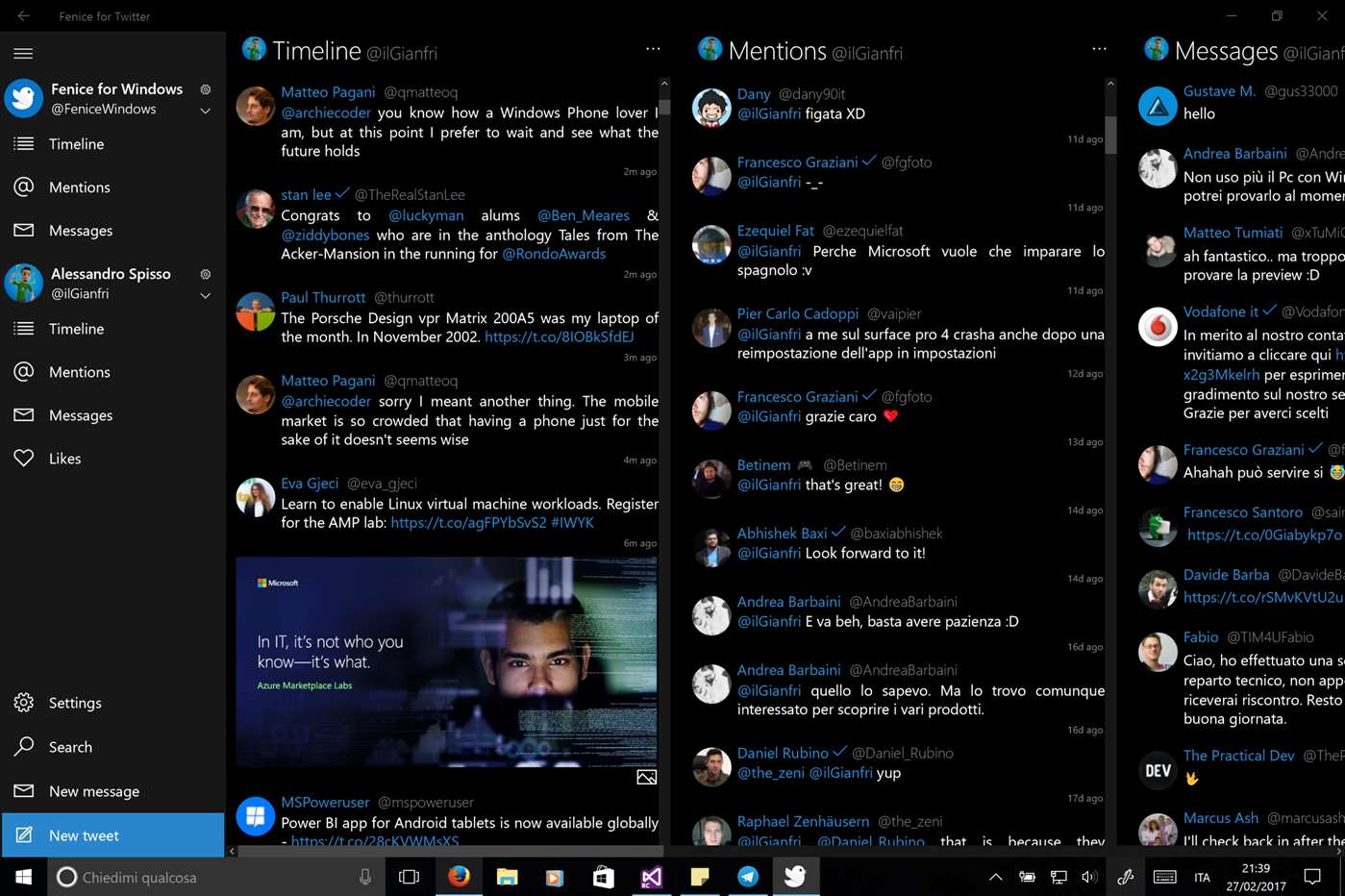
Fenice (рис 1.4) – еще один альтернативный полнофункциональный и мощный Твиттер – клиент, но уже исключительно для Windows 10.

Рисунок 1.4 – приложение Fenice для Windows 10

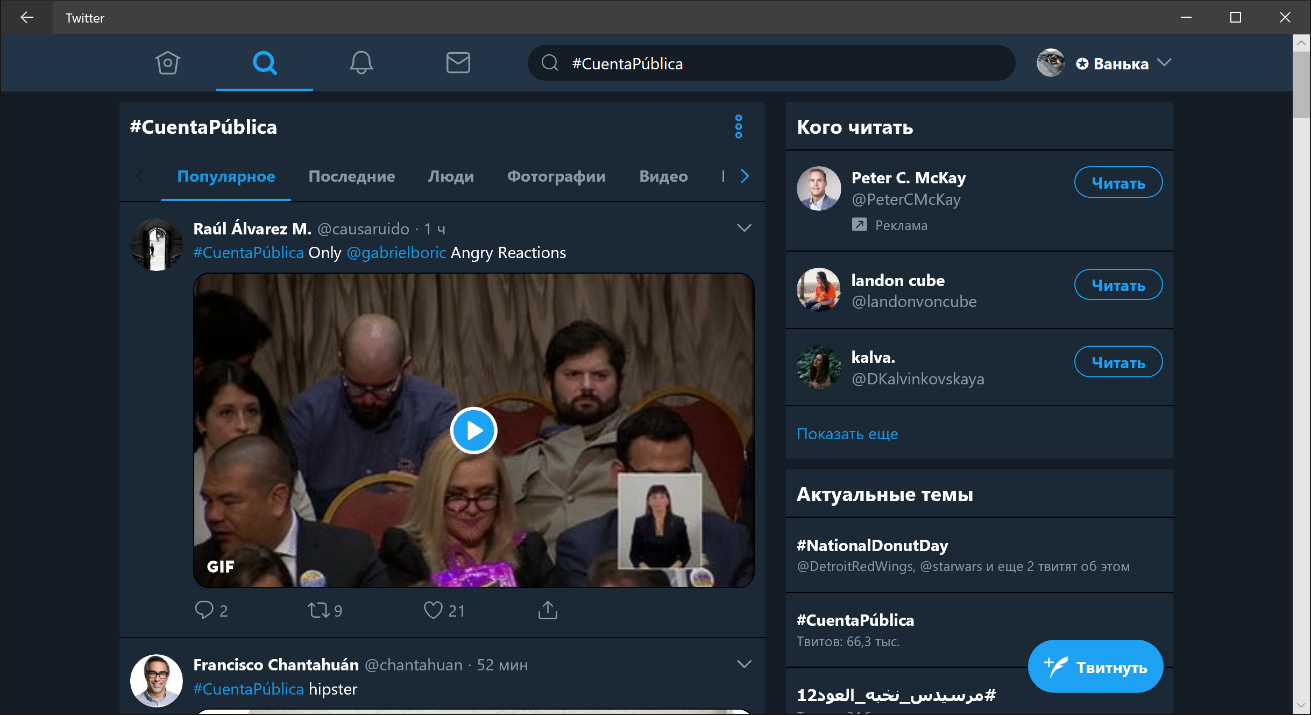
Также у компании Twitter есть свое полнофункциональное настольное приложение для Windows 10.

Рисунок 1.5 – официальное приложение Twitter для Windows 10

Все вышеперечисленные приложения имеют свой особенный уникальный и приятный пользовательский интерфейс. Реализованы все основные и много других дополнительных функций социальной сети Твиттер.

* 1. **Формирование требований к проектируемому программному средству**

Целью данной работы является разработка и создание ПС, позволяющего пользователю заходить в свой Твиттер – аккаунт и использовать основные функции социальной сети.

При выполнении данного курсового проекта, поставлены следующие задачи:

1. реализация соединения с серверами Твиттера;
2. реализация получения данных с серверов;
3. реализация отправки данных на сервера;
4. реализация интерфейса для корректного отображения полученных с серверов данных;
5. реализация интерфейса для осуществления корректной отправки данных серверам;

**2 РАЗРАБОТКА АЛГОРИТМА**

**2.1 Анализ требований к программному средству**

В результате анализа программного средства были составлены следующие функциональные требования:

1. отправление твитов от аккаунта пользователя;
2. просмотр твитов пользователя;
3. просмотр новостной ленты (твитов людей, на которых подписан пользователь);
4. отображение изображения профиля пользователя;
5. отображение основной информации аккаунта (ник пользователя, количество твитов, подписчиков и подписок);
6. корректная обработка возможных ошибок различного рода и вывод информации о них;
7. возможность выхода из аккаунта;
8. дружелюбный пользовательский интерфейс;

**2.2** **Проектирование**

В данной работе проектирование сводится к описанию классов для работы с сервером и для реализации основной формы. Также были описаны классы для создания дополнительных форм и класс для загрузки шрифтов.

Структура классов проекта представлена на рисунке 2.1

Рисунок 2.1 – структура классов в проекте

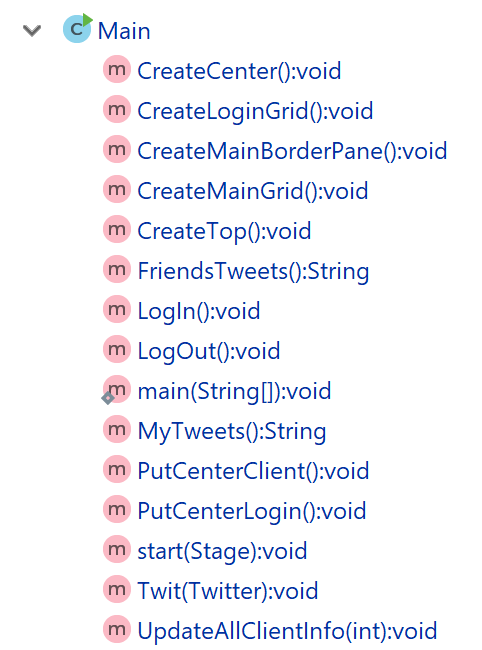
Класс Main является основным. В нем происходит инициализация всех элементов окна и обработка событий. На рисунке 2.2 представлена структура методов класса Main, который предназначен для выполнения всех возможных операций, поддерживаемых ПС.

Рисунок 2.2 – структура методов класса Main

В данном программном средстве не было необходимости проектировать сложные базы данных, так как основные манипуляции происходят с сервером.

**2.3 Реализация**

На рисунке 2.3 представлена реализация получения объекта twitter типа Twitter, который описан в подключаемой библиотеке. При наличии подключения к сети интернет и ввода корректного Pin – кода аккаунта, который пользователь получит с официального сайта при разрешении данному приложению пользоваться его данными, произойдет инициализация переменной twitter, дальнейшие взаимодействия с которой помогут мне реализовать получение твитов, их отправку, а также получение различной информации аккаунта.



Рисунок 2.3 – класс TwitterLogin

**3 ОБОСНОВАНИЕ ТЕХНИЧЕСКИХ ПРИЕМОВ**

**ПРОГРАММИРОВАНИЯ**

Написание Твиттер - клиента на Java я начал с изучения JavaFX. JavaFX — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing.

Среди возможностей этой платформы можно отметить: кроссплатформенность, поддержка каскадных таблицей стилей, поддержка анимации компонентов, возможность работы и отображение 3D графики, поддержка тачей и сенсоров и очень многое другое.

Для удобства работы с Твиттер - сервером мною была использована библиотека Twitter4J версии 4.0.5.

Twitter4J - неофициальная библиотека Java для API Twitter. С Twitter4J можно легко интегрировать приложение Java с сервисом Twitter. Работает на любой платформе Java версии 5 или более поздней версии. Twitter API 1.1 совместима. Встроена поддержка OAuth и gzip.

**4** **ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

Тестированием программного обеспечения является процесс исследования, испытания программного продукта, имеющий две различные цели:

* показать разработчикам и заказчикам, что программа соответствует всем необходимым требованиям;
* выявить некорректное поведение программы.

Тестированием программы называется этап, на котором проверяется программа, а именно как ведёт себя она на большом объёме входных данных, включая заведомо неправильные. Существует множество принципов организации тестирование, далее приведены основные из них.

1. важная часть тестирования – описание ожидаемых результатов работы программа. Это следует делать для того, чтобы потом можно было быстро выявить наличие или отсутствие ошибки;
2. досконально изучать результаты каждого теста, чтобы не упустить ошибку в работе программы;
3. необходимо тщательно подбирать данные не только для правильных входных данных, но и для неправильных;
4. при анализе полученных тестов следует проверять, не выполняет ли программа лишних действий;
5. следует сохранять полученные результаты тестирования программы для повышения качества повторного тестирования;
6. для проведения тестирования программы следует выделять достаточное количество необходимых ресурсов;
7. всегда помнить, что тестирование является творческим процессом и не относится к нему как к рутинному занятию.

Существует несколько видов тестирования:

* функциональное тестирование;
* структурное тестирование;
* регрессивное тестирование;
* альфа-тестирование;
* бэта-тестирование;
* автоматизированное тестирование.

Функциональное тестирование – тип тестирование на соответствие программного обеспечения всем функциональным требованиям. Обычно программное обеспечение тестируется как черный ящик. При таком виде тестирования на вход мы подаём какие-либо данные и получаем результат, при этом не зная, как они обрабатываются.

Структурное тестирование – тестирование кода на предмет логики работы программы. Для полной реализации структурного тестирования необходимо покрыть:

* все строки кода;
* все ветви программы;
* все пути в программе.

Регрессивное тестирование направленно на обнаружении ошибок в уде ранее протестированных участках исходного кода.

Альфа-тестирование представляет собой имитацию реальной работы с системой штатными разработчиками, с привлечением огромного числа реальных пользователей системы. Данный вид тестирования проходит исключительно на стороне разработчика программного обеспечения.

Бэта-тестирование представляет реальную работу программного обеспечения на стороне заказчика с привлечение огромного числа реальных пользователей системы. Программное обеспечение при бэта-тестировании обычно характеризуется урезанным функционалом и относительно большим числом дефектов.

Автоматизированное тестирование – тестирование программного обеспечения с использованием специального программного обеспечения. Для его реализации фактически требуется разработать новое программное средство. Так существуют готовые программы для данного вида тестирования, такие как SilkperFormer и RFT. SilkperFormer используют для нагрузочного тестирования, а RFT для функционального тестирования.

Так же при тестировании программного обеспечения используются следующие понятия: тестирование, надёжность, дефект, жизненный цикл дефекта. Тестирование – процесс анализа программного обеспечения, связанный с выявлением дефектов. Надёжность – свойство объекта сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения. Дефект – отклонение поведения программного продукта от ожидаемого. Жизненный цикл дефекта – период времени от выявления данного дефекта до момента окончания разработки программного обеспечения.

Важную роль в проведении тестирования играет ведение соответствующей документации. Она включает в себя план тестирования и отчёт о дефекте. План тестирования – документ, отображающий последовательность действий, необходимых для проверки программного обеспечения на соответствие требуемому качеству. Отчёт о баге – документ, содержащий данные и несоответствие поведения программного обеспечения каким-либо соответствующим требованиям и алгоритмам для повторения дефекта.

В программе предусмотрена обработка следующих исключительных ситуаций:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Тест | Ожидаемый результат | Полученный результат |
| 1 | Нажатие кнопки Log in при отсутствии подключения к сети интернет | Получение сообщения об ошибке |  |
| 2 | Ввод некорректного Pin – кода | Получение сообщения об ошибке Pin - кода |  |
| 3 | Отправка пустого твита | Получение сообщения о пустом твите |  |
| 4 | Попытка ввода твита размером больше 280 символов | Размер вводимого текста ограничен 280 символами, больше ввести не является возможным |  |
| 5 | Нажатие кнопки Tweet при отсутствии подключения к сети интернет | Получение сообщения об ошибке |  |
| 6 | Нажатие кнопки Refresh при отсутствии подключения к сети интернет | Получение сообщения об ошибке |  |
| 7 | Нажатие кнопки Refresh более 15 раз за 15 минут (ограничение Твиттера) | Получение сообщения об ошибке |  |
| 8 | Попытка ввода твита, который был недавно отправлен (ограничение Твиттера) | Получение сообщения об ошибке |  |

Таблица 4.1 – таблица тестирования исключительных ситуаций

**5** **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

После запуска приложения, пользователь видит окно, представленное на рисунке 5.1.

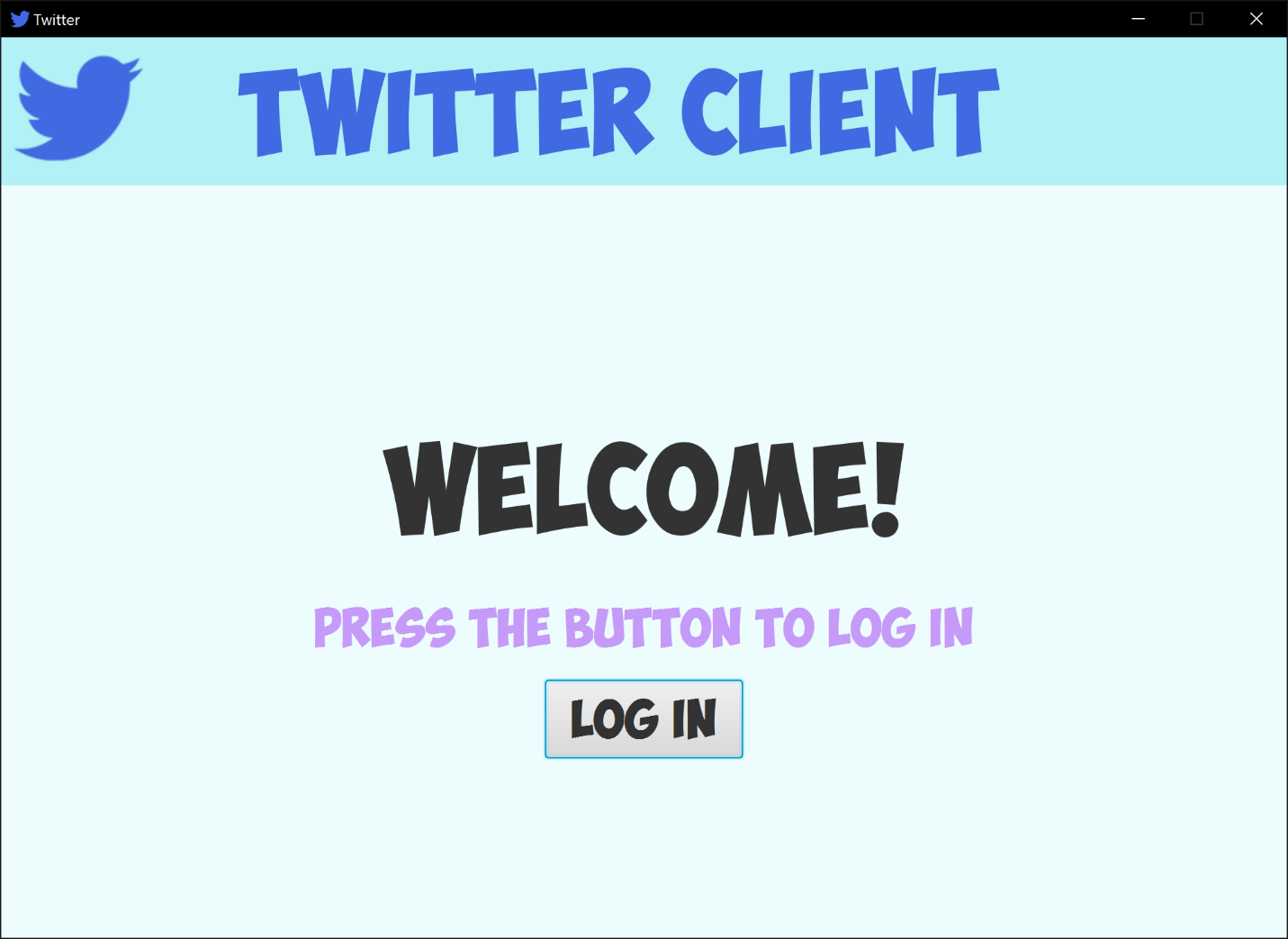


Рисунок 5.1 − Начальный экран

После нажатия на кнопку Log in открывается официальный сайт Твиттера (рис 5.2) и просит подтверждения для использования аккаунта пользователя в приложении.

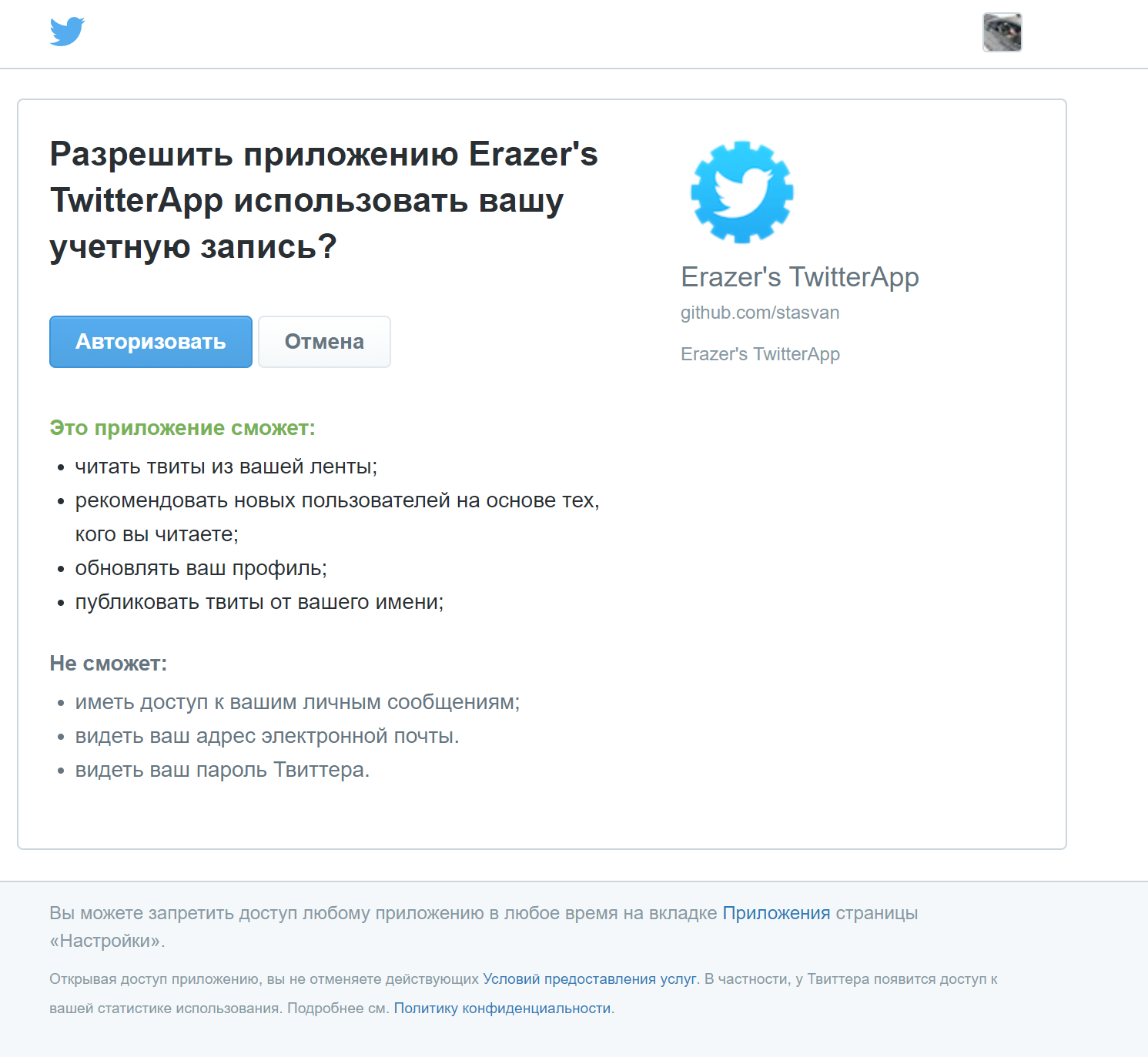


Рисунок 5.2 − Запрос подтверждения авторизации

После подтверждения пользователь получает Pin – код (рис 5.3), который надо ввести в приложение для дальнейшей работы (рис 5.4).

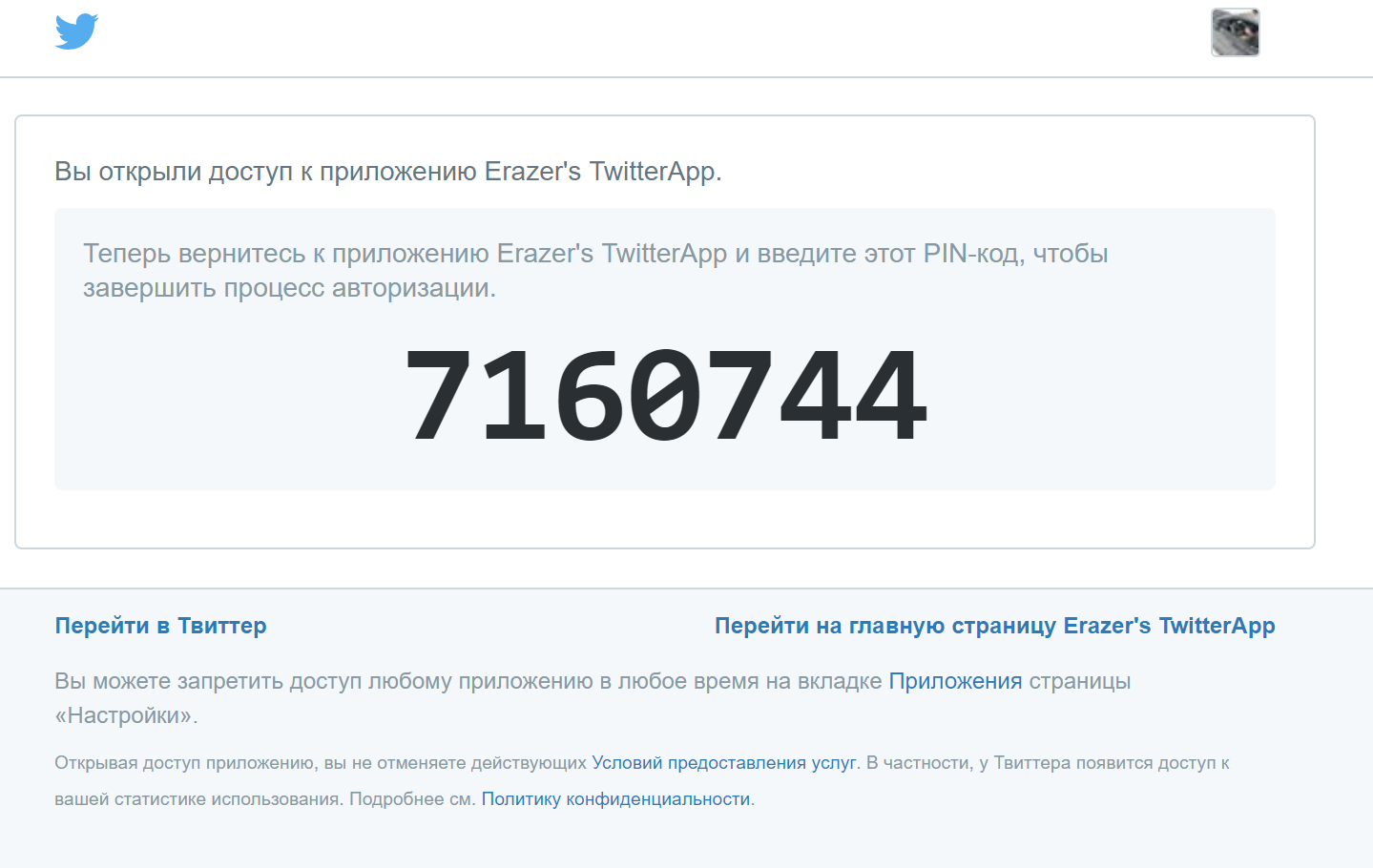


Рисунок 5.3 − полученный Pin - код

Рисунок 5.4 − поле для ввода Pin – кода

 После ввода Pin - кода происходит вход в аккаунт пользователя и основное окно изменяется (рис 5.5).

Рисунок 5.5 − Основное окно после входа в аккаунт

Пользователь может отправлять твиты, просматривать последние 20 твитов друзей и последние 20 собственных твитов, а также обновлять данные.

При необходимости можно выйти из аккаунта нажав на кнопку Log out. В таком случае пользователь попадет на начальный экран.

**ЗАКЛЮЧЕНИЕ**

Мной было создано программное средство — Твиттер - клиент, позволяющее осуществлять доступ и взаимодействие с аккаунтом пользователя в социальной сети Твиттер.

В процессе создания данного проекта мной были изучены некоторые особенности языка программирования высокого уровня Java, были получены навыки работы с этим языком, а также с JavaFX. Также были получены и закреплены навыки в объектно-ориентированном программировании.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 01.01.1992. – М.: Изд-во стандартов, 1991.

[2] Сайт - «Oracle» [Электронный ресурс]. — Режим доступа: <https://www.oracle.com/index.html>

[3] Сайт - «Википедия» [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org

[4] Сайт - «Twitter4j» [Электронный ресурс]. – Режим доступа: http://twitter4j.org/en/index.html

# 

**Приложение А**

**Исходный код программы**

**Main.java**

**package** sample.mainForm;  
  
**import** javafx.application.Application;  
  
**import** javafx.geometry.Insets;  
**import** javafx.geometry.Pos;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.Label;  
**import** javafx.scene.control.TextArea;  
**import** javafx.scene.image.Image;  
**import** javafx.scene.image.ImageView;  
**import** javafx.scene.layout.\*;  
**import** javafx.scene.paint.Color;  
**import** javafx.scene.text.Font;  
**import** javafx.stage.Stage;  
**import** javafx.scene.control.Button;  
**import** sample.messageBoxes.AlertBox;  
**import** sample.fonts.Fonts;  
**import** sample.twitterLogin.TwitterLogin;  
**import** twitter4j.\*;  
  
**import** java.io.FileInputStream;  
**import** java.io.FileNotFoundException;  
  
**public class** Main **extends** Application{  
  
 **private final int WINDOW\_HEIGHT** = 700, **WINDOW\_WIDTH** = 1000;  
 **private** BorderPane **mainBorderPane**;  
 **private** VBox **loginGrid**;  
 **private** VBox **showBox**;  
 **private** HBox **tweetTweet**;  
 **private** Label **whatToShow**;  
 **private** GridPane **mainGrid**;  
 **private** BorderPane **topPane**;  
 **private boolean centerLogin** = **true**;  
 **private** Button **buttonLogOut**;  
 **private** Button **buttonLogIn**;  
 **private** Label **labelLogIn**;  
 **private** Label **welcom**;  
 **private** Label **nick**;  
 **private** Label **tweetsCount**;  
 **private** Label **followersCount**;  
 **private** Label **followingCount**;  
 **private** Button **tweetButton**;  
 **private** Button **showFriendsTweets**;  
 **private** Button **showMyTweets**;  
 **private** Button **refresh**;  
 **private** ImageView **selectedImage**;  
 **private** TextArea **tweetArea**;  
 **private** TextArea **tweetsArea**;  
 **private** Twitter **twitter**;  
 **private** Font **font**;  
 **private** User **user**;  
 **private** String **myTweets**;  
 **private** String **friendsTweets**;  
 **private int id**;  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
  
 @Override  
 **public void** start(Stage window) {  
 window.setTitle(**"Twitter"**);  
 window.setResizable(**false**);  
 window.centerOnScreen();  
 window.getIcons().add(**new** Image(**"file:src/pics/logo.png"**));  
 **font** = Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 20);  
 CreateCenter();  
 CreateTop();  
 **mainBorderPane** = **new** BorderPane();  
 CreateMainBorderPane();  
 Scene scene = **new** Scene(**mainBorderPane**, **WINDOW\_WIDTH**, **WINDOW\_HEIGHT**);  
  
 window.setScene(scene);  
 window.show();  
 }  
  
 **private void** CreateCenter() {  
 CreateLoginGrid();  
 *//CreateMainGrid();* }  
  
 **private void** CreateLoginGrid() {  
 *//CENTER* **loginGrid** = **new** VBox();  
 **loginGrid**.setAlignment(Pos.***CENTER***);  
 **loginGrid**.setPadding(**new** Insets(60,0,10,0));  
  
 **buttonLogIn** = **new** Button(**"Log in"**);  
 **buttonLogIn**.setFont(Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 30));  
 **buttonLogIn**.setOnAction(e -> LogIn());  
  
 Font fontLabelLogIn = Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 30);  
 **labelLogIn** = **new** Label(**"Press the button to Log in"**);  
 **welcom** = **new** Label(**"Welcome!"**);  
 **welcom**.setPadding(**new** Insets(0,0,30,0));  
 **labelLogIn**.setPadding(**new** Insets(10,0,20,0));  
 **labelLogIn**.setFont(fontLabelLogIn);  
 fontLabelLogIn = Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 70);  
 **welcom**.setFont(fontLabelLogIn);  
 **labelLogIn**.setTextFill(Color.*web*(**"#C59AF9"**));  
  
 **loginGrid**.getChildren().addAll(**welcom**, **labelLogIn**, **buttonLogIn**);  
 **loginGrid**.setStyle(**"-fx-background-color: #EDFCFC;"**);  
 *//END CENTER* }  
  
 **private void** CreateMainGrid() {  
 **mainGrid** = **new** GridPane();  
 **mainGrid**.setPrefSize(100,100);  
 **mainGrid**.setPadding(**new** Insets(10,10,10,10));  
 **mainGrid**.setVgap(20);  
 **mainGrid**.setHgap(20);  
 **mainGrid**.setStyle(**"-fx-background-color: #EDFCFC;"**);  
 **tweetTweet** = **new** HBox();  
  
 **tweetArea** = **new** TextArea();  
 **tweetArea**.setPrefColumnCount(45);  
 **tweetArea**.setPrefRowCount(4);  
 **tweetArea**.setFont(**new** Font(15));  
 **tweetArea**.setFocusTraversable(**true**);  
 **final int** LIMIT = 280;  
 **tweetArea**.lengthProperty().addListener((observable, oldValue, newValue) -> {  
 **if** (newValue.intValue() > oldValue.intValue()) {  
 **if** (**tweetArea**.getText().length() >= LIMIT) {  
 **tweetArea**.setText(**tweetArea**.getText().substring(0, LIMIT));  
 }  
 }  
 });  
 **tweetArea**.setWrapText(**true**);  
 *//mainGrid.add(tweetArea, 0, 0);* VBox butVBox = **new** VBox();  
 butVBox.setPadding(**new** Insets(0, 10, 0, 10));  
 **tweetButton** = **new** Button(**"Tweet"**);  
 **tweetButton**.setOnAction(e -> Twit(**twitter**));  
 **tweetButton**.setPadding(**new** Insets(5, 20, 5, 20));  
 **tweetButton**.setFont(**font**);  
 **nick** = **new** Label(**""**);  
 Font nickFont = **new** Font(20);  
 **nick**.setFont(nickFont);  
 **nick**.setTextFill(Color.*web*(**"#C59AF9"**));  
 **nick**.setPadding(**new** Insets(0, 0, 5, 0));  
 nickFont = **new** Font(15);  
 **tweetsCount** = **new** Label(**""**);  
 **tweetsCount**.setFont(nickFont);  
 **tweetsCount**.setTextFill(Color.*web*(**"#C59AF9"**));  
 **tweetsCount**.setPadding(**new** Insets(0, 0, 5, 0));  
 **followersCount** = **new** Label(**""**);  
 **followersCount**.setFont(nickFont);  
 **followersCount**.setTextFill(Color.*web*(**"#C59AF9"**));  
 **followersCount**.setPadding(**new** Insets(0, 0, 5, 0));  
 **followingCount** = **new** Label(**""**);  
 **followingCount**.setFont(nickFont);  
 **followingCount**.setTextFill(Color.*web*(**"#C59AF9"**));  
 **followingCount**.setPadding(**new** Insets(0, 0, 5, 0));  
  
 butVBox.getChildren().addAll(**nick**,**tweetsCount**, **followersCount**, **followingCount**, **tweetButton**);  
 *//mainGrid.add(butVBox, 1, 0);* **tweetTweet**.getChildren().addAll(**tweetArea**, butVBox);  
 **mainGrid**.add(**tweetTweet**, 0, 0);  
  
 **selectedImage** = **new** ImageView();  
  
 **try** {  
 **user** = **twitter**.showUser(**twitter**.getId());  
 } **catch** (TwitterException e) {  
 e.printStackTrace();  
 }  
 String path = **user**.getOriginalProfileImageURL();  
 Image logo = **new** Image(path);  
 **selectedImage**.setImage(logo);  
 **selectedImage**.setFitHeight(149);  
 **selectedImage**.setFitWidth(149);  
 **mainGrid**.add(**selectedImage**, 1, 0);  
  
 **tweetsArea** = **new** TextArea();  
 **tweetsArea**.setPrefColumnCount(60);  
 **tweetsArea**.setPrefRowCount(18);  
 **tweetsArea**.setFont(**new** Font(15));  
 **tweetsArea**.setFocusTraversable(**true**);  
 **tweetsArea**.setWrapText(**true**);  
 **tweetsArea**.setEditable(**false**);  
 **mainGrid**.add(**tweetsArea**, 0, 1);  
  
 **showBox** = **new** VBox();  
 **showBox**.setSpacing(15);  
 **showBox**.setPrefWidth(150);  
  
 **whatToShow** = **new** Label(**"What to show:"**);  
 Font fontWhatToShow = **new** Font(20);  
 **whatToShow**.setFont(fontWhatToShow);  
 **whatToShow**.setTextFill(Color.*web*(**"#C59AF9"**));  
 **showFriendsTweets** = **new** Button(**"Friends'\ntweets"**);  
 **showFriendsTweets**.setOnAction(e -> {  
 **id** = 1;  
 **tweetsArea**.setText(**friendsTweets**);  
 });  
 **showFriendsTweets**.setFont(**font**);  
 **showFriendsTweets**.setMinWidth(**showBox**.getPrefWidth());  
 **showFriendsTweets**.setPadding(**new** Insets(10,10,10,10));  
 **showMyTweets** = **new** Button(**" My\ntweets"**);  
 **showMyTweets**.setOnAction(e -> {  
 **id** = 0;  
 **tweetsArea**.setText(**myTweets**);  
 });  
 **showMyTweets**.setFont(**font**);  
 **showMyTweets**.setMinWidth(**showBox**.getPrefWidth());  
 **showMyTweets**.setPadding(**new** Insets(10,10,10,10));  
 **refresh** = **new** Button(**"Refresh"**);  
 **refresh**.setOnAction(e -> UpdateAllClientInfo(**id**));  
 **refresh**.setFont(**font**);  
 **refresh**.setPadding(**new** Insets(10,10,10,10));  
 **refresh**.setMinWidth(**showBox**.getPrefWidth());  
  
 **showBox**.getChildren().addAll(**whatToShow**,**showMyTweets**,**showFriendsTweets**, **refresh**);  
 **mainGrid**.add(**showBox**, 1, 1);  
  
 *//mainGrid.setGridLinesVisible(true);* }  
  
 **private void** Twit(Twitter twitter) {  
  
 **if** (twitter != **null**) {  
  
 **try** {  
 **if** (!**tweetArea**.getText().trim().equals(**""**)) {  
 twitter.updateStatus(**tweetArea**.getText());  
 **tweetArea**.setText(**""**);  
 AlertBox.*display*(**"Message"**, **"Tweeted!"**, **font**);  
 } **else** {  
 AlertBox.*display*(**"Message"**, **"Tweet is empty!"**, **font**);  
 }  
 } **catch** (TwitterException e) {  
 AlertBox.*display*(**"Message"**, **"Error"**, **font**);  
 }  
 *//UpdateAllClientInfo();* }  
  
 }  
  
 **private void** CreateTop() {  
 *//TOP* **topPane** = **new** BorderPane();  
  
 HBox topCenterMenu = **new** HBox();  
 topCenterMenu.setPadding(**new** Insets(15,0,10,70));  
 topCenterMenu.setSpacing(10);  
 topCenterMenu.setStyle(**"-fx-background-color: #B2F1F5;"**);  
 Label text = **new** Label(**"Twitter Client"**);  
 Font fontBig = Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 65);  
 text.setFont(fontBig);  
 text.setTextFill(Color.*web*(**"#4169E1"**));  
 topCenterMenu.getChildren().addAll(text);  
  
 HBox topLeftMenu = **new** HBox();  
 topLeftMenu.setPadding(**new** Insets(5,5,10,10));  
 *//topLeftMenu.setSpacing(10);* topLeftMenu.setStyle(**"-fx-background-color: #B2F1F5;"**);  
 **try** {  
 ImageView selectedImage = **new** ImageView();  
 Image logo = **new** Image(**new** FileInputStream(**"src/pics/logo.png"**));  
 selectedImage.setImage(logo);  
 topLeftMenu.getChildren().add(selectedImage);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
  
 VBox topRightMenu = **new** VBox();  
 topRightMenu.setPadding(**new** Insets(55,30,5,0));  
 *//topRightMenu.setSpacing(10);* topRightMenu.setStyle(**"-fx-background-color: #B2F1F5;"**);  
 **buttonLogOut** = **new** Button(**"Log out"**);  
 **buttonLogOut**.setFont(**font**);  
 **buttonLogOut**.setTextFill(Color.*web*(**"#4169E1"**));  
 **buttonLogOut**.setVisible(**false**);  
 **buttonLogOut**.setOnAction(e -> LogOut());  
  
 topRightMenu.getChildren().addAll(**buttonLogOut**);  
  
 **topPane**.setRight(topRightMenu);  
 **topPane**.setCenter(topCenterMenu);  
 **topPane**.setLeft(topLeftMenu);  
 *//END TOP* }  
  
 **private void** CreateMainBorderPane() {  
  
 **mainBorderPane**.setCenter(**loginGrid**);  
 **mainBorderPane**.setTop(**topPane**);  
  
 }  
  
 **private void** LogOut() {  
 **if** (!**centerLogin**) {  
 **twitter** = **null**;  
 PutCenterLogin();  
 System.***out***.println(**"log out"**);  
 **centerLogin** = **true**;  
 **buttonLogOut**.setVisible(**false**);  
 }  
  
 }  
  
 **private void** LogIn() {  
 **buttonLogIn**.setDisable(**true**);  
 **twitter** = TwitterLogin.*LogIn*();  
 **if** (**twitter** != **null**) {  
 PutCenterClient();  
 System.***out***.println(**"log in"**);  
 *//AlertBox.display("Message", "You are logged in", font);* UpdateAllClientInfo(0);  
 **centerLogin** = **false**;  
 **buttonLogOut**.setVisible(**true**);  
 } **else** {  
 System.***out***.println(**"log in fail"**);  
 }  
 **buttonLogIn**.setDisable(**false**);  
 }  
  
 **private void** UpdateAllClientInfo(**int** id) {  
  
 **try** {  
 **user** = **twitter**.showUser(**twitter**.getId());  
 } **catch** (TwitterException e) {  
 e.printStackTrace();  
 }  
 **nick**.setText(**user**.getScreenName());  
 **tweetsCount**.setText(**"Tweets: "** + **user**.getStatusesCount());  
 **followersCount**.setText(**"Followers: "** + **user**.getFollowersCount());  
 **followingCount**.setText(**"Following: "** + **user**.getFriendsCount());  
 **myTweets** = MyTweets();  
 **friendsTweets** = FriendsTweets();  
 **if** (id == 0) {  
 **tweetsArea**.setText(**myTweets**);  
 } **else if** (id == 1){  
 **tweetsArea**.setText(**friendsTweets**);  
 }  
  
 }  
  
 **private** String MyTweets() {  
 ResponseList<Status> list;  
 StringBuilder strB = **new** StringBuilder(**"Your tweets:\n-------------------------------------------------------------------------------------------------------------------------------\n"**);  
 **try** {  
 list = **twitter**.getUserTimeline();  
 **for** (Status status : list) {  
 strB.append(**"@"**).append(status.getUser().getScreenName()).append(**" - "**).append(status.getText()).append(**"\n"**).append(**"-------------------------------------------------------------------------------------------------------------------------------\n"**);  
 }  
 } **catch** (TwitterException e) {  
 AlertBox.*display*(**"Message"**,**"User's rate\nlimit exceeded"**, **font**);  
 **return myTweets**;  
 }  
  
 **return** strB.toString();  
 }  
  
 **private** String FriendsTweets() {  
 ResponseList<Status> list;  
 StringBuilder strB = **new** StringBuilder(**"Friends' tweets:\n-------------------------------------------------------------------------------------------------------------------------------\n"**);  
 **try** {  
 list = **twitter**.getHomeTimeline();  
 **for** (Status status : list) {  
 strB.append(**"@"**).append(status.getUser().getScreenName()).append(**" - "**).append(status.getText()).append(**"\n"**).append(**"-------------------------------------------------------------------------------------------------------------------------------\n"**);  
 }  
 } **catch** (TwitterException e) {  
 AlertBox.*display*(**"Message"**,**"Friends' rate\nlimit exceeded"**, **font**);  
 **return friendsTweets**;  
 }  
  
 **return** strB.toString();  
 }  
  
 **private void** PutCenterClient() {  
 CreateMainGrid();  
 **mainBorderPane**.setCenter(**mainGrid**);  
 }  
  
 **private void** PutCenterLogin() {  
 CreateCenter();  
 **mainBorderPane**.setCenter(**loginGrid**);  
 }  
  
}

**TwitterLogin.java**

**package** sample.twitterLogin;  
  
**import** javafx.scene.text.Font;  
**import** sample.fonts.Fonts;  
**import** sample.messageBoxes.AlertBox;  
**import** sample.messageBoxes.ConfirmPinBox;  
**import** twitter4j.Twitter;  
**import** twitter4j.TwitterException;  
**import** twitter4j.TwitterFactory;  
**import** twitter4j.auth.AccessToken;  
**import** twitter4j.auth.RequestToken;  
**import** twitter4j.conf.ConfigurationBuilder;  
  
**import** java.awt.\*;  
**import** java.net.URI;  
  
**public class** TwitterLogin {  
  
 **public static** Twitter LogIn() {  
  
 Font font = Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 20);  
 ConfigurationBuilder cb = **new** ConfigurationBuilder();  
 cb.setDebugEnabled(**true**)  
 .setOAuthConsumerKey(**"0Cj4B7jX58aBX9weNgZJ2ymks"**)  
 .setOAuthConsumerSecret(**"IcTBCn40cdBaDp3npx0QbGE7jR127Qajd5lNJbi8GrDQooTflW"**);  
  
 TwitterFactory tf = **new** TwitterFactory(cb.build());  
 Twitter twitter = tf.getInstance();  
  
 **try** {  
 RequestToken requestToken = twitter.getOAuthRequestToken();  
 System.***out***.println(**"Got request token."**);  
  
 System.***out***.println(**"Enter the PIN"**);  
 String pin;  
 Desktop desktop = Desktop.*isDesktopSupported*() ? Desktop.*getDesktop*() : **null**;  
 pin = ConfirmPinBox.*display*(font, desktop, requestToken);  
  
 **if** (pin.equals(**"-1"**)) {  
 **return null**;  
 }  
 **try** {  
 Integer.*parseInt*(pin);  
 } **catch** (NumberFormatException e) {  
 AlertBox.*display*(**"Message"**,**"Pin is not validate"**, font);  
 **return null**;  
 }  
  
 **try** {  
 twitter.getOAuthAccessToken(requestToken, pin);  
 } **catch** (TwitterException ignored) {  
 AlertBox.*display*(**"Message"**,**"Pin is not validate"**, font);  
 **return null**;  
 }  
 System.***out***.println(**"Got access token."**);  
 } **catch** (IllegalStateException | TwitterException ie) {  
 AlertBox.*display*(**"Message"**, **"No Internet connection"**, font);  
 **return null**;  
 }  
 System.***out***.println(**"ready to twit"**);  
 **return** twitter;  
 }  
}

**AlertBox.java**

**package** sample.messageBoxes;  
  
**import** javafx.geometry.Pos;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.Label;  
**import** javafx.scene.image.Image;  
**import** javafx.scene.layout.VBox;  
**import** javafx.scene.paint.Color;  
**import** javafx.scene.text.Font;  
**import** javafx.stage.Modality;  
**import** javafx.stage.Stage;  
  
  
**public class** AlertBox {  
 **public static void** display(String title, String message, Font font) {  
 Stage window = **new** Stage();  
 window.initModality(Modality.***APPLICATION\_MODAL***);  
 window.getIcons().add(**new** Image(**"file:src/pics/logo.png"**));  
 window.setTitle(title);  
 window.setResizable(**false**);  
 window.centerOnScreen();  
  
 Label label = **new** Label();  
 label.setFont(font);  
 label.setTextFill(Color.*web*(**"#C59AF9"**));  
 label.setText(message);  
 Button closeButton = **new** Button(**"Ok"**);  
 closeButton.setFont(font);  
 closeButton.setOnAction(e -> window.close());  
  
 VBox layout = **new** VBox();  
 layout.setStyle(**"-fx-background-color: #EDFCFC;"**);  
 layout.getChildren().addAll(label, closeButton);  
 layout.setAlignment(Pos.***CENTER***);  
  
 Scene scene = **new** Scene(layout, 300, 150);  
 window.setScene(scene);  
 window.showAndWait();  
 }  
}

**ConfirmPinBox.java**

**package** sample.messageBoxes;  
  
**import** javafx.geometry.Insets;  
**import** javafx.geometry.Pos;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.TextField;  
**import** javafx.scene.image.Image;  
**import** javafx.scene.layout.VBox;  
**import** javafx.scene.paint.Color;  
**import** javafx.scene.text.Font;  
**import** javafx.stage.Modality;  
**import** javafx.stage.Stage;  
**import** javafx.scene.control.Label;  
**import** sample.fonts.Fonts;  
**import** twitter4j.auth.RequestToken;  
  
**import** java.awt.\*;  
**import** java.net.URI;  
  
  
**public class** ConfirmPinBox {  
  
 **private static** String *pin* = **"-1"**;  
  
 **public static** String display(Font font, Desktop desktop, RequestToken requestToken){  
 *pin* = **"-1"**;  
 Stage window = **new** Stage();  
 window.getIcons().add(**new** Image(**"file:src/pics/logo.png"**));  
 window.initModality(Modality.***APPLICATION\_MODAL***);  
 window.setTitle(**"Twitter"**);  
 window.setResizable(**false**);  
 window.centerOnScreen();  
 **if** (desktop != **null** && desktop.isSupported(Desktop.Action.***BROWSE***)) {  
 **try** {  
 desktop.browse(URI.*create*(requestToken.getAuthorizationURL())); } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 **final int** LIMIT = 7;  
 TextField tf = **new** TextField();  
 tf.lengthProperty().addListener((observable, oldValue, newValue) -> {  
 **if** (newValue.intValue() > oldValue.intValue()) {  
 *// Check if the new character is greater than LIMIT* **if** (tf.getText().length() >= LIMIT) {  
 *// if it's 11th character then just setText to previous  
 // one* tf.setText(tf.getText().substring(0, LIMIT));  
 }  
 }  
 });  
 tf.setText(**""**);  
 tf.setFont(font);  
  
 Label label = **new** Label(**"Enter pin"**);  
 label.setTextFill(Color.*web*(**"#C59AF9"**));  
 label.setFont(font);  
  
 Button btn = **new** Button(**"Go"**);  
 btn.setFont(font);  
 btn.setOnAction(e -> {  
 *pin* = tf.getText();  
 window.close();  
 });  
  
 font = Fonts.*LoadFont*(**"src/fonts/ObelixPro.ttf"**, 26);  
 tf.setFont(font);  
  
 VBox layout = **new** VBox();  
 layout.setPadding(**new** Insets(10,30,10,30));  
 layout.setStyle(**"-fx-background-color: #EDFCFC;"**);  
 layout.getChildren().addAll(label, tf, btn);  
 layout.setAlignment(Pos.***CENTER***);  
  
 Scene scene = **new** Scene(layout, 250, 150);  
 window.setScene(scene);  
 window.showAndWait();  
 window.setOnCloseRequest(e -> {  
 *pin* = **"-1"**;  
 window.close();  
 });  
  
 **return** *pin*;  
 }  
}

**Fonts.java**

**package** sample.fonts;  
  
**import** javafx.scene.text.Font;  
  
**import** java.io.File;  
**import** java.io.FileInputStream;  
**import** java.io.FileNotFoundException;  
  
**public class** Fonts {  
  
 **public static** Font LoadFont(String path, **int** size){  
  
 **try** {  
 **return** Font.*loadFont*(**new** FileInputStream(**new** File(path)), size);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 **return null**;  
 }  
  
 }  
  
}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | | Дополнительные сведения | | | | |
|  | | | | Текстовые документы | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
| БГУИР КР 1–40 01 01 523 ПЗ | | | | Пояснительная записка | | | | | 36 с. | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | | Графические документы | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
| ГУИР 651005 523 СП | | | | Общая схема программы | | | | | Формат А1 | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  | | | |  | | | | |  | | | | |
|  |  |  |  |  | БГУИР КР 1-40 01 01 523 Д1 | | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Программное средство «Twitter - клиент» |  | | Лист | | | | Листов |
| Разраб. | | Стаселович И.А. |  | 02.06.2018 | Т |  | | |  |  | 36 |
| Пров. | | Балашко А.О. |  | 02.06.2018 | Кафедра ПОИТ  гр. 651005 | | | | | | |
|  | |  |  |  |
|  | |  |  |  |