Software Architecture Document

Version <1.0>

Applicativi Touch in ambiente Microsoft- 2017

Daniele Zuppello

Indice

Sommario

[1. Introduzione 3](#_Toc492368890)

[1.1 Obiettivo 3](#_Toc492368891)

[1.2 Definizioni, Acronimi, e Abbreviazioni 3](#_Toc492368892)

[1.3 Riferimenti 3](#_Toc492368893)

[2. Rappresentazione architetturale 4](#_Toc492368894)

[3. Fattori architetturali 4](#_Toc492368895)

[3.1 Piattaforme 4](#_Toc492368896)

[4. Decisioni architetturali 5](#_Toc492368897)

[5. Logical View 5](#_Toc492368898)

[5.1 Overview 5](#_Toc492368899)

[5.2 Namespace significativi 5](#_Toc492368900)

[5.2.1 Demo.BusinessLogic 5](#_Toc492368901)

[5.2.2 Demo.Controls 5](#_Toc492368902)

[5.2.3 Demo.Manipulation 6](#_Toc492368903)

[5.2.4 Demo.Migrations 6](#_Toc492368904)

[5.2.5 Demo.Themes 6](#_Toc492368905)

[5.2.6 Demo 6](#_Toc492368906)

[6. Use-Case View 7](#_Toc492368907)

[6.1 Realizzazioni di Use-Case 7](#_Toc492368908)

[7. Data View 8](#_Toc492368909)

# Introduzione

Il sistema operativo Windows non è più un’esclusiva dei computer che potremmo definire “classici”.

E’ ormai dalla versione 8 che esso è approdato anche su tablet, smartphone ed in generale su device che richiedono, od hanno la possibilità, di essere usati tramite uno schermo touch. Questo documento si inserisce in una attività di studio per cercare di comprendere come sviluppare applicativi touch in ambienti Microsoft

## Obiettivo

L’obiettivo dato dallo Stackholder è quello di riuscire a creare degli applicativi touch utilizzabili in parallelo da più persone.

Questi applicativi dovranno essere formati da una serie di widget con cui l’utente potrà interagire muovendoli e ruotandoli sullo schermo.

Il Widget sarà quindi l’unità base del sistema con la quale l’utente potrà interagire.

## Definizioni, Acronimi, e Abbreviazioni

|  |  |
| --- | --- |
| Widget | Unità base di interazione tra l’utente e l’applicativo. Fa parte della UI |
| Sistema o Applicativo | Con il termine "sistema" si indica il software, oggetto di questo documento. |
| Stakeholder | Lo stakeholder è il committente del sistema. |
| UWP | Universal Windows Platform, ovvero una tipologia di applicativi utilizzabili su qualsiasi device con installato un sistema operativo Windows 10 |
| UI | User Interface, l’interfaccia utente |
| Namespace | Equivalente dei package Java. Sostanzialmente il namespace definisce uno scope dove definire classi (o altri namespace) che logicamente appartengono ad uno stesso gruppo di funzionalità |

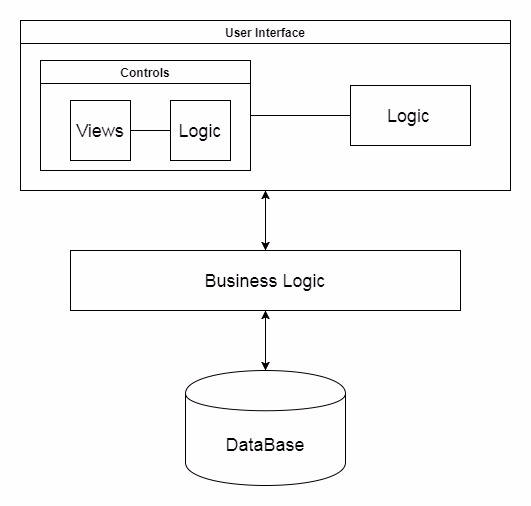
## Riferimenti

Documento SRS (sempre dello stesso progetto) v 1.0

<https://docs.microsoft.com/en-us/windows/uwp> (Documentazione ufficiale Microsoft)

<https://github.com/Microsoft/Windows-universal-samples> (Esempi ufficiali Microsoft)

# Rappresentazione architetturale



# Fattori architetturali

Lo stackeholder ha espressamente richiesto che il sistema fosse sviluppato pensando a quattro precisi fattori architetturali:

1. Flessibilità
2. Semplicità di sviluppo
3. Riusabilità del codice
4. Manutenibilità del codice

Si è inoltre cercato di mantenere l’interfaccia utente il più semplice possibile per rendere l’applicativo di facile utilizzo.

## Piattaforme

L’applicazione è implementata con il linguaggio di programmazione C# e con il framework .NETCore

Per la parte grafica si è invece usato il linguaggio XAML, un dialetto dell’XML

Essendo l’applicativo di tipologia UWP potrà essere utilizzato su qualsiasi device su cui è installato Windows 10.

# Decisioni architetturali

Per ottenere un grado di manutenibilità soddisfacente è necessario separare il più possibile parti di codice che implementano funzionalità differenti. È dunque opportuno dividere i dati, la logica di business, e la presentazione delle informazioni. In questo modo ogni singolo componente potrà essere testato separatamente in quanto dovrà adempiere ad un solo particolare scopo slegato dalla logica d’insieme del sistema

Per gli stessi motivi anche il codice sarà facilmente riutilizzabile in quanto esso risulterà indipendente dall’applicativo in sé dovendo adempiere solo ad un determinato compito (quale può essere l’accesso ad un database) ed essendo quindi facilmente inseribile in contesti diversi.

Anche la flessibilità e la semplicità di sviluppo sono accentuate dalla suddetta divisione poiché si può andare a modificare una determinata funzionalità di uno strato senza andare ad impattare sugli altri strati (a patto, ovviamente, di non cambiare le firme dei metodi che si vanno a modificare ma solo le funzionalità). Ciò permette di dividere i task con più facilità e di far lavorare in parallelo più sviluppatori.

# Logical View

## Overview

L'applicazione è divisa vari strati che si occupano rispettivamente delle interazioni con l'utente, della logica funzionale e della comunicazione con il database.

Per evitare di complicare tale struttura l’N-esimo strato può comunicare solo con il strato in posizione N+1 o N-1.

Inoltre è stata prestata particolare attenzione anche a rimuovere le dipendenze tra il primo strato dedicato all'interfaccia grafica e il strato di logica funzionale in modo da poter rendere quanto più facile ed indipendente dal resto dell'applicazione lo sviluppo di nuove interfacce grafiche.

## Namespace significativi

### Demo.BusinessLogic

Il namespace business logic contiene, come da nome, la logica di business dell’applicazione.

Questo namespace contiene due classi: Model e DBUtility. La prima, Model, contiene la rappresentazione sotto forma di classe del Database mentre la seconda, DBUtility, contiene i metodi utili ad accedere al database attraverso il model

### Demo.Controls

Il namespace controls contiene le classi di code behind dei widget dove sono state definite le proprietà modificabili degli stessi permettendo così di adattarli a situazioni e pagine differenti.

Questo namespace contiene inoltre la classe MyControl, sottoclasse di Control ovvero la classe di base per tutti gli elementi di interfaccia utente, dalla quale ereditano tutti gli altri widget. Questa classe permette quindi di definire dei metodi utili a tutti i controllori, evitando così inutili ripetizioni di codice nei widget.

### Demo.Manipulation

Questo namespace contiene la classe ManipulationInputProcessor, ovvero la classe adibita al controllo degli input touch da parte dell’utente.

In essa sono gestiti una serie di eventi, relativi al tocco dello schermo, grazie ai l’utente riuscirà ad interagire con i widget muovendoli e ruotandoli

### Demo.Migrations

Questo namespace contiene le classi, generate automaticamente da Visual Studio, che permettono la migrazione del modello che abbiamo creato nel namespace BusinessLogic in un reale database.

### Demo.Themes

Questo namespace contiene la parte grafica dei widget che abbiamo definito.

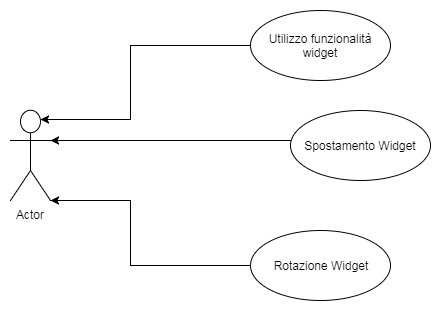
La parte grafica è scritta nel linguaggio XAML

### Demo

Contiene le pagine dell’applicativo con il relativo codebehind

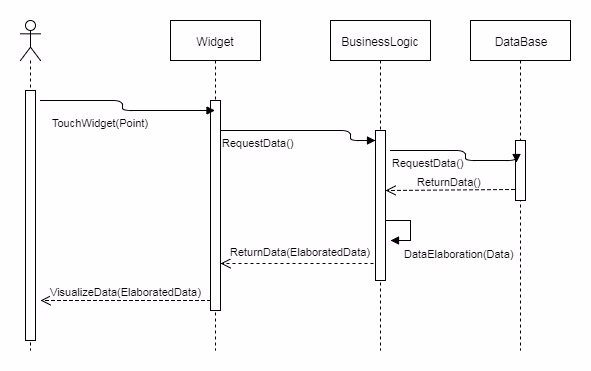
# Use-Case View

Use case implementati nell’applicativo



## Realizzazioni di Use-Case

Di seguito viene riportato il sequence diagram rappresentate l’interazione dell’utente con il widget:



# Data View

A fini puramente dimostrativi è stato creato un semplice database. Tale database è formato da una singola tabella contenente dei libri descritti dai seguenti campi: un intero valido come identificativo univoco, una stringa per il titolo, una stringa per l’autore ed una stringa per l’url dell’immagine di copertina.

Come database è stato scelto SQLite, una libreria che implementa un DBMS SQL. Grazie ad SQLite si riesce a simulare, tramite un semplice file di testo, un reale database. Ciò rende SQLite una buona scelta per applicativi mobile (essendo il file di testo in generale leggero ed efficiente per un quantitativo di dati non troppo grande) e per una veloce prototipazione (come nel nostro caso).

Per accedere al database dall’applicativo si è scelto di sfruttare EntityFramework, ovvero un framework ORM open source proprietario di Microsoft con il quale è possibile far comunicare due sistemi incompatibili attraverso l’uso della programmazione ad oggetti creando un “oggetto database virtuale” .