1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO

Dipartimento di Scienze Pure e Applicate
Corso di Laurea in Informatica Applicata

Tesi di Laurea

COS'È BLAZOR?

Relatore:
Chiar.mo Prof. Emanuele Lattanzi

Candidato:
Francesco Belacca

Anno Accademico 2018-2019

A tutti quelli che mi hanno detto almeno una volta di non mollare.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Contesto

A seguito della crescita esponenziale del web in questo secolo e dell'abituarsi di coloro che ne usufruiscono ad un livello grafico sempre migliore e ad una esperienza mano a mano più interattiva e vicina all'utente medio, i siti web e le tecnologie utilizzate si sono adattati per permettere uno sviluppo sempre più rapido di codice più facilmente testabile e mantenibile.

Di conseguenza nel frontend si sono susseguiti una serie di framework a partire da JQuery nel 2006, che per primo si è occupato di risolvere il problema della compatibilità tra browsers, permettendo ai developers di scrivere una volta, e poter eseguire su tutti i browsers.

AngularJS nel 2010 è stato il primo MVC framework ad offrire in un unico pacchetto, two-way data binding, dependency injection, routing facilitato e altri strumenti utili per rendere più standard lo sviluppo nel frontend [1]. Dopo la riscrittura di questo framework nel 2013 che è diventato Angular 2 (e recentemente Angular) senza mantenere retrocompatibilità e senza offrire un modo preciso per migrare alla nuova versione agli utilizzatori di AngularJS, React, un nuovo framework più leggero e modulare sviluppato dagli sviluppatori di Facebook, ha preso il posto di Angular come framework più utilizzato.

Vue infine é il terzo dei principali framework che ha provato a prendere piede proponendo una versione intermedia tra il fortemente opinionato Angular e il piú flessibile React.

Oltre a questi, ciascuno con la propria semantica, organizzazione logica dei folder, spesso una CLI dedicata, ad un developer frontend viene solitamente richiesto di conoscere HTML, CSS e Javascript su cui si basano poi i vari framework.

Oltre a Javascript, se si vuole scrivere degli unit test facilmente mantenibili, bisogna conoscere TypeScript(specialmente se si utilizza Angular, che rende il suo utilizzo obbligatorio) e degli altri framework che facilitino i test(Enzyme, Karma + Jasmine, ...).

1.2 Problema

I continui cambiamenti nei molti framework utilizzati, la diversità degli strumenti stessi tra loco, che spesso realizzano in modo diverso tutti la stessa cosa(framework concorrenti), rendono specialmente per un junior developer molto ampia la curva di apprendimento e lo studio necessario, per essere anche solo operativo.

Oltretutto un developer ad oggi finisce per essere costretto a scegliere se diventare uno sviluppatore frontend o backend, dato che rimanere al passo e aggiornarsi già in solo uno di questi due campi richiede tempo e non è scontato ad esempio che venga concesso di poterlo fare in orario lavorativo, pur essendo fondamentale.

É quindi chiaro che per tutti i web developer, e specialmente per una figura mista spesso identificata con il titolo "Full-Stack Developer", ci sia una continua ricerca del modo per rendere le proprie competenze quanto piú trasversali possibile tra frontend e backend, anche in termini di tecnologia utilizzata.

1.3 Linguaggi General-Purpose

Microsoft ha reso nel tempo il framework .NET e le sue implementazioni(.NET Core, .NET Framework e Mono) utilizzabili nei vari linguaggi supportati, C#, F# e VB.

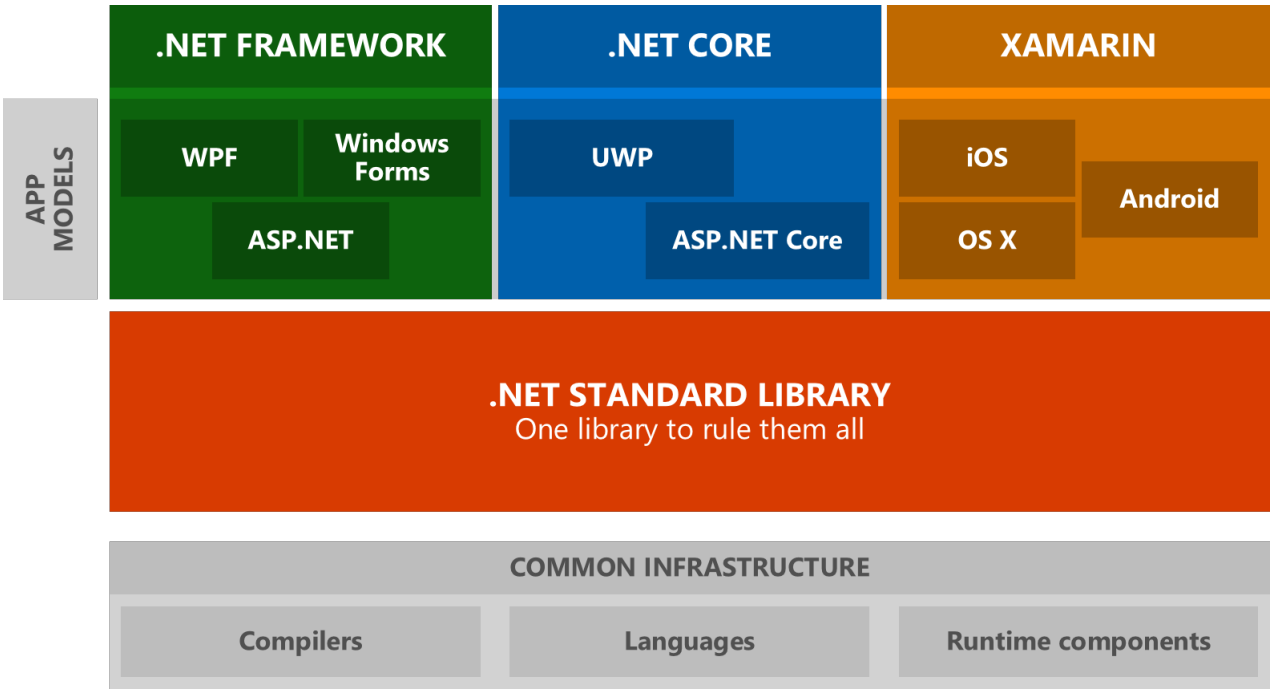

Figura 1.1: Implementazioni di .NET

Scrivendo ad esempio in C# è possibile sviluppare vari tipi di applicazioni, ma se si decide di sviluppare codice per un applicazione client web ad oggi si è ancora costretti a scrivere utilizzando Javascript e un suo framework se si vuole essere veloci nello sviluppo e scrivere codice mantenibile in team piú grandi, come nel mondo enterprise.

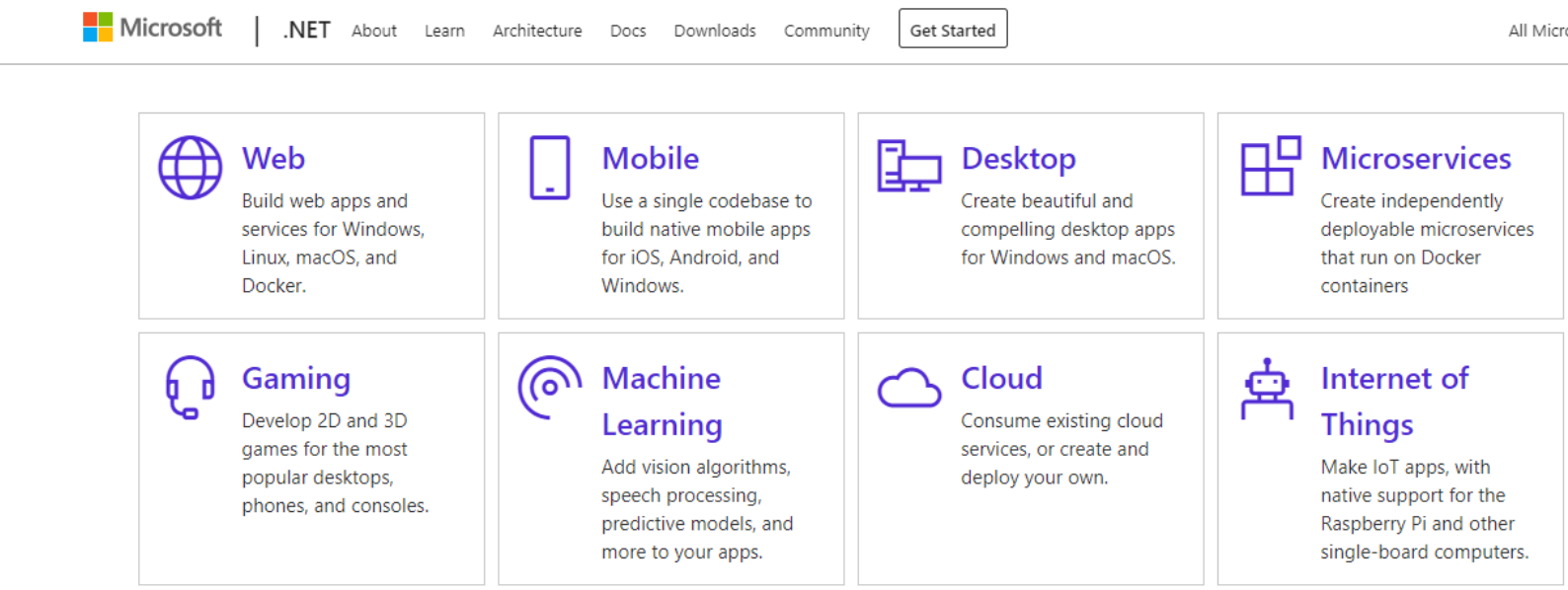

Figura 1.2: Possibilitá di .NET

Giá con Razor[2], Microsoft ha provato a permettere la generazione di codice HTML e CSS in modo dinamico utilizzando C#, ma è utilizzabile solo lato server e quindi ad esempio la cattura di un evento client side come il click di un utente su un bottone senza contattare il server non è gestibile utilizzando il solo C#.

Blazor é quindi la versione successiva(Web-Razor) che permette ai developer di gestire anche gli eventi client-side, direttamente in C# come se questo fosse effettivamente ció che viene eseguito lato client, mentre in reatá l'esecuzione lato client cambia a seconda del modello scelto, come poi vedremo piú nel dettaglio.

Capitolo 2

Modelli e Funzionamento

2.1 Blazor Server

2.2 Blazor WebAssembly

2.3 Blazor PWA

2.4 Blazor Hybrid

2.5 Blazor Native

Capitolo 3

Pro e Contro

3.1 Pro

3.2 Contro

Capitolo 4

Conclusioni

4.1 Conclusioni

4.2 Futuro del progetto

Bibliografia

[1] M. Wanyoike, Disponibile: https://blog.logrocket.com/history-of-frontend-frameworks.

[2] Disponibile: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-3.0&tabs=visual-studio.

Ringraziamenti

Vorrei ringraziare il professor Lattanzi per avermi aiutato a scrivere questa tesi, i miei genitori per avermi permesso di procedere facendo di testa mia su un percorso a loro estraneo, e la mia ragazza Ivana per aver creduto in me e nelle mie capacità anche quando non l'ho fatto io.