ALMA MATER STUDIORUM – UNIVERSITA'

DI BOLOGNA CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE

INFORMATICHE

Find to play Android App

Relazione finale in

Mobile Web Design

Relatore:

Dott. Mirko Ravaioli

Presentata da:

Andrea Caselli

Sessione II

Anno Accademico 2014/2015

INDICE

INDICE DELLE FIGURE

1 INTRODUZIONE

Negli ultimi dieci anni con l'avvento degli smartphone abbiamo assistito ad un radicale cambiamento nella tecnologia passando da dispositivi specifici come cellulari e palmari ad altri, gli smartphone, che ne unissero le capacità, integrandole assieme ad un'altra serie di funzionalità come l'accesso a internet, riproduzione multimediale, fotocamera, geolocalizzazione e tutto ciò che da esso è derivato.

La penetrazione globale degli smartphone è ormai salita al di sopra del 60%, e le vendite di tablet dovrebbero superare quelle di computer desktop nell'anno in corso. In questo nuovo ecosistema, sette minuti di utilizzo dei telefoni cellulari su otto vengono spesi all'interno delle applicazioni, nel solo 2014 l'utilizzo delle mobile app è cresciuto del 76%.

Essendo il mobile il vero regno delle app, il dato di base è che lo spostamento verso la mobilità è sempre più accentuato. Connessioni a banda larga mobile (cioè tecnologie 3G e 4G) rappresentavano poco meno del 40% del totale delle connessioni alla fine del 2014, ma entro il 2020 si stima che queste aumenteranno fino a quasi il 70% del totale.

Questa migrazione è anche facilitata dalla maggiore disponibilità ed accessibilità economica degli smartphone, e anche da una maggiore qualità e copertura del segnale offerta degli operatori telefonici.

Nel corso del XVI Evento Annuale di Ericsson, l'azienda ha rivelato che da quest'anno, i cellulari sono più degli esseri umani: stiamo infatti parlando di oltre 7,2 miliardi di telefonini contro 7 miliardi di esseri umani.
Non solo il numero di cellulari è ormai superiore a quello degli uomini, ma buona parte dei dispositivi sono smartphone. Si calcola che nel 2016 la quota degli smartphone possa arrivare al 50% del totale.

1.1 SISTEMI A CONFRONTO

A questi dispositivi in costante ascesa bisogna associare un sistema operativo che stia al passo con l'evolversi delle esigenze dell'utenza. I sistemi che attualmente dominano il mercato sono principalmente tre:

- iOS
- Android
- Windows Phone

1.1.1 IOS

È stato il primo dei tre sistemi ad aver raggiunto il pubblico in veste simile a quella attuale. iOS è stato ed è tuttora il sistema presente su tutti i dispositivi mobile targati Apple.

È ottimizzato per i dispositivi su cui viene fatto girare, tanto che difficilmente capiterà di vedere blocchi, malfunzionamenti, comportamenti strani o rallentamenti nella risposta dell'interfaccia: tutto è pensato e sviluppato per funzionare perfettamente in sinergia all'hardware.

1.1.2 ANDROID

Android è il sistema operativo mobile di Google. Attualmente è il più diffuso al mondo ed è considerato uno dei maggiori concorrenti di iOS. È per la quasi totalità free e open source. Samsung, HTC, Motorola e molti altri grandi costruttori stanno usando Android nei loro device.

Questo sistema operativo viene usato su dispositivi con fasce di prezzo e caratteristiche molto varie, inoltre il fatto che sia open source, fa sì che diventi possibile per le case produttrici personalizzare il sistema.
Tutto ciò rallenta o rende impossibili gli aggiornamenti portando ad avere device con versioni del sistema piuttosto datate.

Tale diversità è il motivo della frammentazione del mondo Android che porta chi sviluppa le app a dover tener conto di queste di differenze allungando quindi i tempi di sviluppo.

1.1.3 WINDOWS PHONE

Già dal nome si intuisce che il proprietario è Microsoft. Dal 2011 questo è diventato il principale sistema operativo degli smartphone Nokia. Qualche anno più tardi Microsoft acquista il reparto dedicato alla produzione di telefoni da Nokia assicurandosi i diritti del marchio Lumia. Nasce così *Microsoft Mobile.* Questa scelta è stata dettata dalla volontà di Microsoft di costruire un ecosistema attorno ai propri prodotti come fatto da Apple.
Con l'imminente roll out del nuovo sistema operativo mobile Windows 10 Mobile ci si aspettano cambiamenti consistenti da un sistema operativo che è sempre rimasto un po' nell'angolo rispetto ai più famosi concorrenti Android e iOS. Cambiamenti che riguardano sia sviluppatori, andando ad aumentare il poco ricco store di applicazioni, sia per gli utenti; implementando procedure diventate ormai uno standard per l'usabilità di un telefono.

1.2 SCELTA DELLA PIATTAFORMA

Il sistema scelto per lo sviluppo dell'app è Android.

Questa decisione è stata presa in base alla diffusione dei sistemi e a disponibilità tecnologiche.

Nel secondo quadrimestre del 2015, secondo stime fatte dal *International Data Corporation* (IDC), risulta che Android detiene il primato con 82,8% del mercato. Seguono i device iOS con il 13,9%. Fanalino di coda (considerando solo i sistemi operativi mobile più diffusi) Windows phone con il 2,6%.

Per quanto riguarda le esigenze tecnologiche, per lo sviluppo di app per iOS è quasi indispensabile Xcode, disponibile solo per Mac OSX, che ha portato alla scelta di Android come piattaforma di sviluppo dato che

il suo ambiente di sviluppo è disponibile sia per Windows che per Mac che per Linux.

2 OBBIETTIVO DELL'APPLICAZIONE

L'applicazione permette all'utente di cercare un centro sportivo per nome, per città o per provincia. Consente inoltre di visualizzare la disponibilità per ogni singolo campo offerto dalle strutture ed eventualmente di effettuare una prenotazione.

Il centro sportivo renderà disponibili informazioni altrimenti difficilmente reperibili; come gli orari, il numero telefonico, l'indirizzo, i tipi di sport per cui è attrezzato e i campi praticabili (un campo potrebbe rimanere chiuso durante il periodo invernale). Potrà inoltre monitorare velocemente le prenotazioni effettuate sui vari campi.

3 FIND TO PLAY LATO SERVER

Tutte le richieste generate dall'applicazione vengono instradate ad un server. In esso sono memorizzati tutti i dati riguardanti i campi sportivi, gli utenti e il ruolo che ognuno di essi assume. Successivamente verrà specificato meglio cosa si intende per ruolo di un utente, a cosa serve e come viene usato.

I principali punti affrontati per costruire una base solida per questo progetto sono stati i seguenti:

- Progettazione ed implementazione del database ospitato dal Server;
- Configurazione del server per l'accesso da remoto;
- Progettazione ed implementazione dell'applicazione lato Server;

Andiamo ora ad analizzare come sono stati realizzati questi punti.

3.1 PROGETTAZIONE ED IMPLEMENTAZIONE DEL DATABASE

Il sistema che si intende progettare dovrà gestire i centri sportivi (CS), i loro orari, i campi che mettono a disposizione per essere prenotati e il loro costo di affitto; dovrà inoltre gestire gli utenti, i loro ruoli e le loro informazioni personali; infine dovrà occuparsi delle prenotazioni.

I clienti, dei quali si memorizzano nome, cognome, città, immagine profilo, email e password possono cercare un centro sportivo; dei quali viene memorizzato nome, indirizzo, numero di telefono, orari e logo. Ogni CS ha uno o più campi prenotabili. Per ogni campo si memorizzano il costo (che può variare a seconda dell'orario di prenotazione) un flag che indica se il campo è prenotabile o meno e una breve descrizione. Per poter trovare un posto più in fretta un utente può aggiungere un CS ad una lista di preferiti. Il cliente può effettuare prenotazioni per un campo appartenente ad un dato CS. Per effettuare una prenotazione è necessario specificare la data, l'orario di inizio e l'orario di fine.

La sequenza di passi di progettazione è stata divisa in quanto segue:

1. Analisi dei requisiti
2. Progettazione della base di dati
3. Scelta di Software e tecnologie

3.2 ANALISI DEI REQUISITI

La raccolta dei requisiti è inizialmente costituita da specifiche espresse generalmente in linguaggio naturale ed è per questo che risultano spesso ambigue e disorganizzate di conseguenza l'analisi dei requisiti consiste nel chiarimento e nell'organizzazione di tali specifiche.

Cominciamo con l'individuare termini troppo generici o troppo specifici che rendono poco chiaro il concetto. Esplicitiamo le frasi contorte e individuiamo eventuali sinonimi e/o omonimi.

Per la comprensione e la precisazione dei termini usati è utile definire un glossario, che per ogni termine contenga una descrizione, un sinonimo e altri termini, con i quali esiste un legame logico.

Una volta individuate le ambiguità e le imprecisioni, queste verranno eliminate sostituendo i termini non corretti con altri più adeguati. A questo punto modificando le specifiche posso anche decomporre il testo in gruppi di frasi relative ai concetti corretti.

Dati di carattere generale

Il sistema che si intende progettare dovrà gestire i centri sportivi (CS), i loro orari, i campi che mettono a disposizione per essere prenotati e il loro costo di affitto; dovrà inoltre gestire gli utenti, i loro ruoli e le loro informazioni personali; infine dovrà occuparsi delle prenotazioni.

Termini	**Descrizione**	**Sinonimi**
Centro sportivo	Società che gestisce ed affitta campi sportivi	Azienda, CS, Società
Campo	Campo da gioco messo a disposizione dal CS	Oggetto, terreno di gioco
Costo di affitto	Costo orario del campo	Affitto, costo orario
Utente	Colui che utilizza l'applicazione, che sia autenticato o meno al sistema.	Cliente, utilizzatore
Ruolo	Posizione ricoperta dall'utente registrato all'interno del database	Posizione

Dati sui clienti

Per i **clienti** rappresentiamo alcuni dati tra cui il nome, il cognome, l'email, la password, la città e l'immagine di profilo

Dati sui centri sportivi

Per i **centri sportivi** si rappresentano il nome, l'indirizzo, il numero di telefono gli orari di apertura e i campi che ha a disposizione

Dati sui campi

Per ogni **campo da gioco** vengono memorizzati il costo orario (variabile in base alla fascia oraria stabilita dalla società) un flag che serve per indicare se un campo è attualmente usufruibile e prenotabile e una descrizione delle caratteristiche tecniche

Dati sulle prenotazioni

Quando un cliente effettua una prenotazione deve specificare anche la data, l'ora alla quale si intende cominciare a giocare e quando si intende finire la partita.

3.3 PROGETTAZIONE DELLA BASE DI DATI

Il primo punto riguardante la progettazione vera e propria riguarda la progettazione del database, cioè la rappresentazione delle entità coinvolte e dei legami che intercorrono tra di loro.

Il database è un software che permette di strutturare le informazioni e collegarle tra loro nonché di effettuare su di esse operazioni, dette query, che permettono l'interrogazione dello stesso o la manipolazione dei dati oltre che alla modifica della struttura del database stesso.

Il modello attraverso cui vengono strutturate e mantenute le informazioni dipende dal tipo di database in esame.

Nel nostro caso utilizzeremo un database relazionale. Queste funzionalità sono garantite da un software dedicato chiamato DBMS. Il modello utilizzato per rappresentare le basi di dati in questo progetto è il modello E/R, che è anche il modello attualmente più diffuso.

Il modello E/R (Entity/Relation) è un modello per la rappresentazione concettuale dei dati ad un alto livello di astrazione del quale i principali elementi costitutivi sono:

Entità

Rappresentano una classe di oggetti con le stesse proprietà. Ogni istanza rappresenta un singolo oggetto che appartiene alla classe.

Associazione

Rappresenta un legame tra due o più entità

Attributi

Descrivono le entità o gli attributi. Tutti gli oggetti della stessa entità o associazione hanno gli stessi attributi Gli attributi vengono definiti in base al livello di dettaglio con il quale vogliamo descrivere l'entità (o l'associazione) della quale fanno parte.

Identificatori

È costituito da uno o più attributi dell'entità e deve essere unico all'interno della stessa.

In base ai requisiti visti in 3.2, è immediato individuare i concetti principali che possono essere rappresentati da entità nel seguente schema a scheletro.

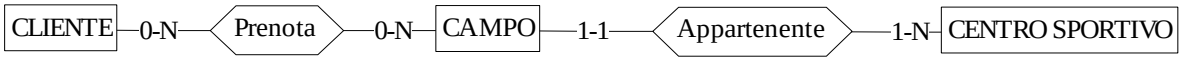


Fig. 1 – Scheletro E/R

A partire dallo schema scheletro effettuiamo una decomposizione dei requisiti fino a giungere al seguente schema ER

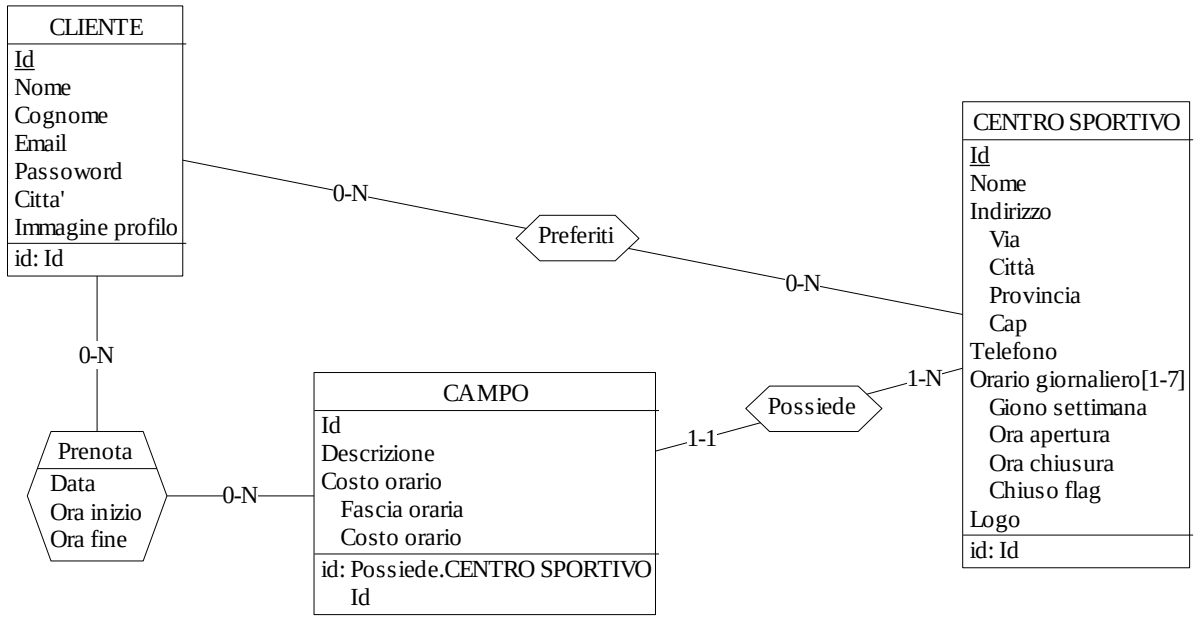


Fig. 2 – Diagramma E/R più dettagliato

Raffinamento dell’entità CENTRO SPORTIVO

Normalizziamo l’entità cliente togliendo gli attributi multi valore, e quelli composti. L’attributo composto Indirizzo viene sostituito da tre attributi distinti (via, città, cap). Mentre l’attributo multi valore è stato sostituito dall’entità distinta “Orario giornaliero”.

Si ha quindi un’entità così trasformata:

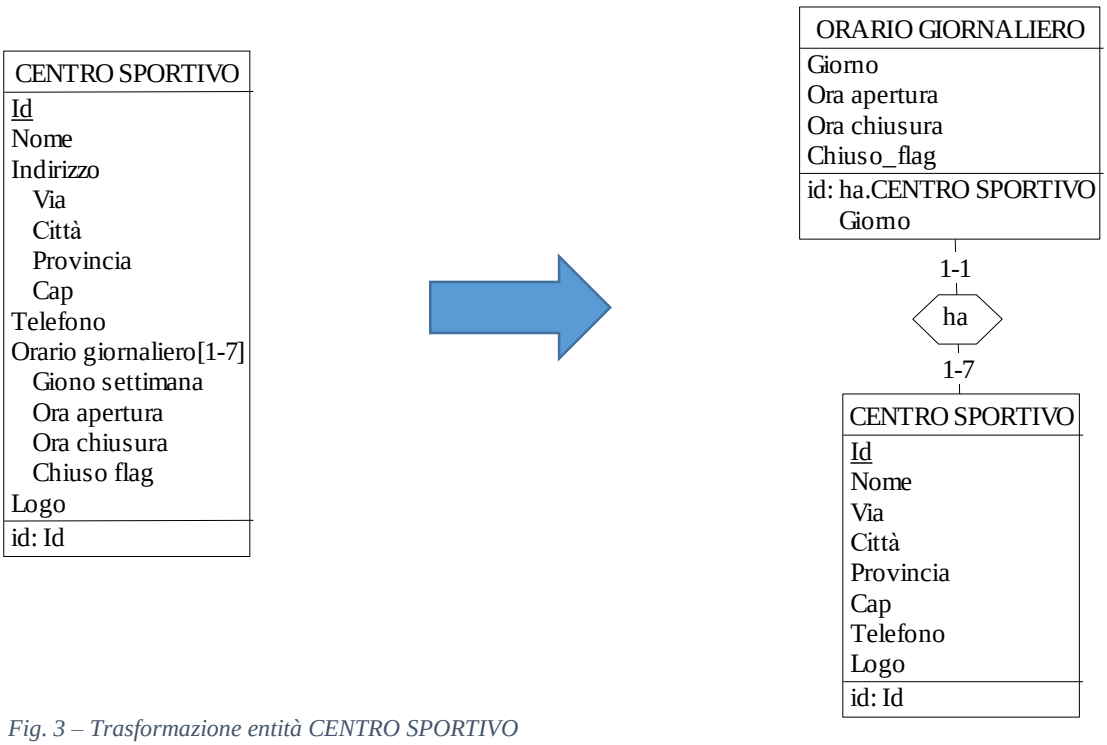


Fig. 3 – Trasformazione entità CENTRO SPORTIVO

Raffinamento dell’entità CAMPO

Come già fatto per i centri sportivi, si sostituisce l’attributo multi valore con un’entità a parte.

Viene inoltre aggiunto un campo boolean che serve a segnalare se il campo è attualmente usufruibile e prenotabile.

Si ha quindi la seguente trasformazione:

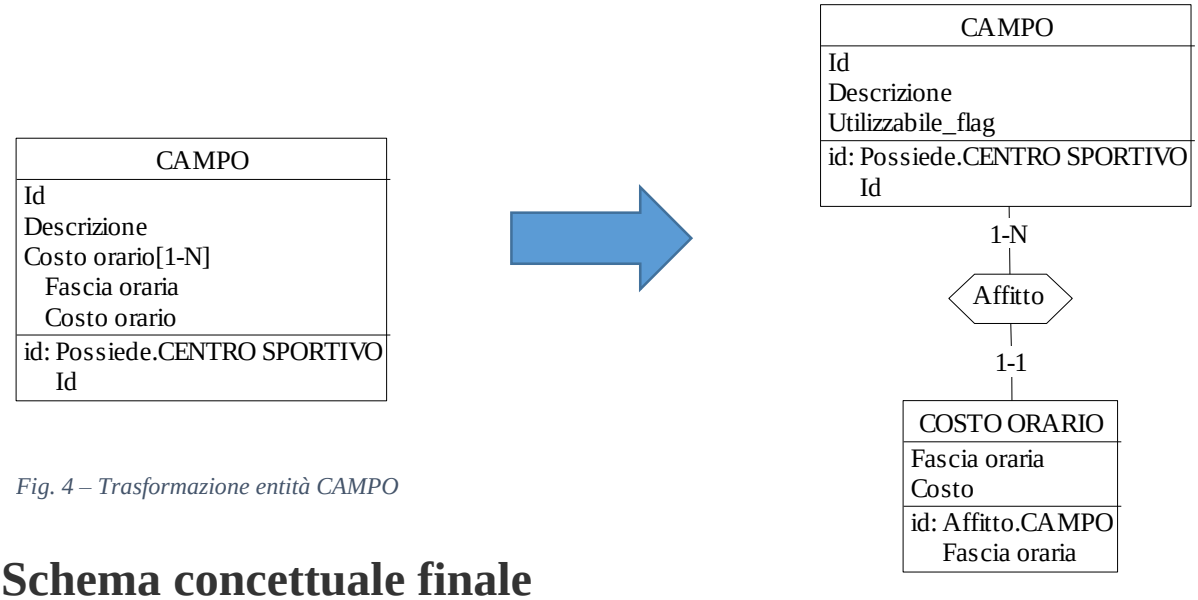


Fig. 4 – Trasformazione entità CAMPO

Schema concettuale finale

Da quanto visto sopra lo schema concettuale finale si presenta in questo modo:

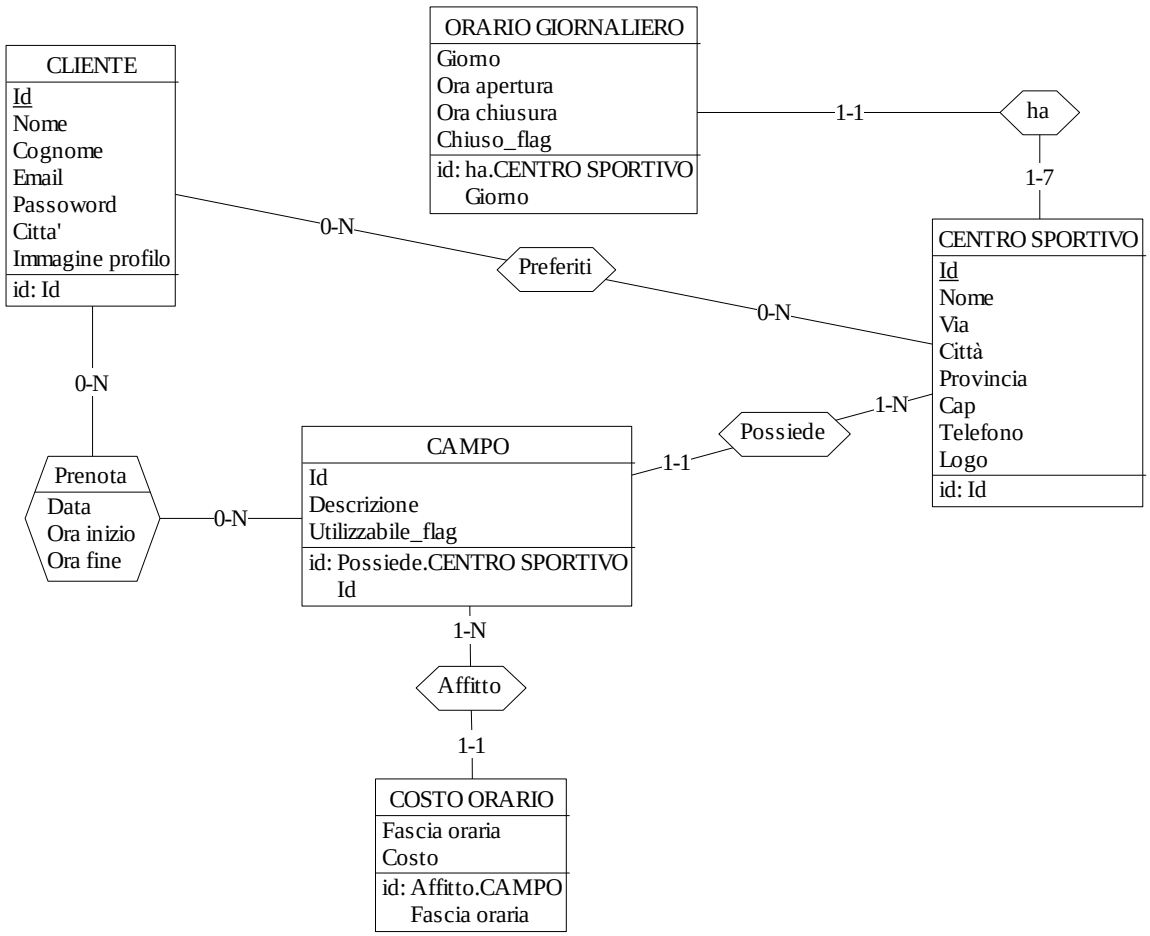


Fig. 5 – E/R finale

Successivamente, tramite una serie di particolari raffinamenti, le entità e le associazioni sono state trasformate in relazioni, e su esse sono stati definiti dei vincoli di integrità, al fine di impostare la struttura del database. Lo schema concettuale è stato quindi trasformato in schema logico.

A questo punto si è passati all'implementazione vera e propria del database.

3.4 SCELTA DI SOFTWARE E TECNOLOGIE

Per lo sviluppo del server è stato scelto di usare il framework ASP.NET Web API 2, il database è gestito attraverso Entity Framework Code First e per la gestione degli user è stato scelto il framework ASP.NET Identity 2.1. Infine, per la gestione e l'autenticazione dell'utente è stato scelto di utilizzare Oauth 2.0 assieme ad un token di tipo JWT (JSON Web Token)

3.4.1 ASP.NET WEB API

Microsoft definisce le Web API come: "Un framework che consente di creare facilmente servizi HTTP in grado di raggiungere un ampio numero di client, inclusi browser e dispositivi mobili. ASP.NET Web API è la piattaforma ideale per creare applicazioni RESTful in .NET Framework".

ASP.NET Web API è stato introdotto come parte di ASP.NET MVC 4 nel 2011; tuttavia, esso ha le sue origini in WCF come WCF Web API. Con l'uscita di MVC 5 è stato rilasciato anche Web api 2 con diverse migliorie rispetto alla sua versione precedente.

Immediatamente può non essere chiara la differenza che c'è tra MVC e Web API. Entrambi utilizzano i controller ma, la seconda, non condivide l'intero Model-View-Controller design di MVC. Entrambi condividono il concetto di mappare le richieste http sui controller ma MVC utilizza un template output e le view per

visualizzare il risultato. Web API restituisce direttamente il modello dell'oggetto come risposta.

Più semplicemente si può dire che MVC è migliore nel accettare form di dati e a generare codice HTML. Mentre le Web API sono perfette per accettare e generare dati di tipo strutturato come JSON e XML.

3.4.2 ADO.NET ENTITY FRAMEWORK

Entity Framework è il framework ORM (object-relational mapping) messo a disposizione dal .NET Framework: il compito svolto dagli ORM è quello di nasconderci il funzionamento di un database relazionale.

Entity Framework è un ORM molto complesso e quindi la sua struttura è composta da diversi componenti.

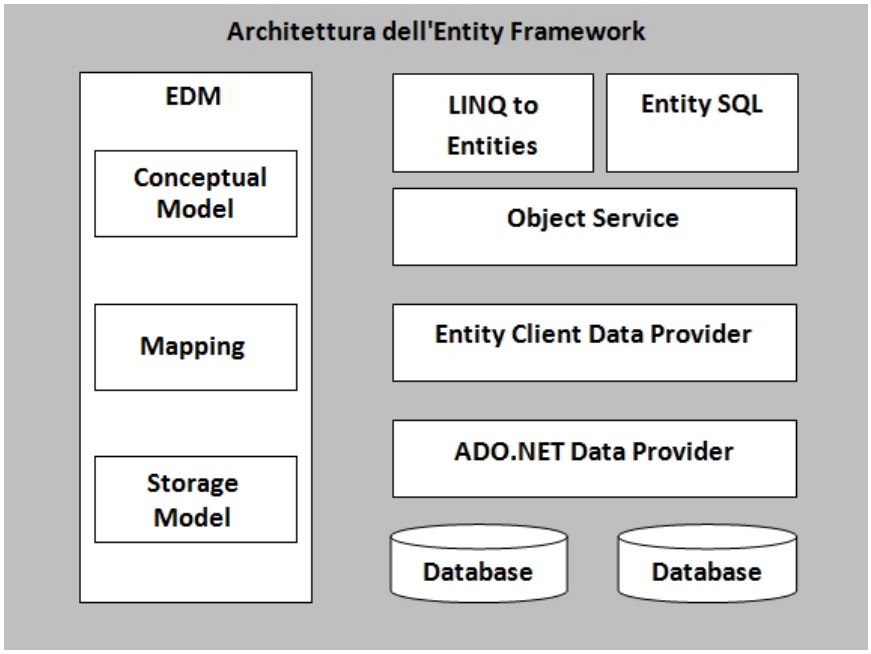


Fig. 6 – Architettura Entity Framework

EDM (Entity Data Model): Si occupa del mapping tra le classi di dominio e le tabelle del database è suddiviso in tre sezioni:

- Conceptual model;
- Mapping;
- Storage model

Conceptual Model: Contiene le classi di dominio e le loro relazioni.

Storage Model: Descrive il database; contiene perciò le tabelle, le views, le stored procedures le loro relazioni con le relative chiavi.

Mapping: descrive il mapping tra le classi e il database. Cioè come il conceptual model è collegato con lo storage model.

LINQ to Entities: è la specializzazione di LINQ che permette di scrivere query verso le classi presenti nel domain model. Sarà poi compito degli strati sottostanti convertire l'expression tree delle query nel linguaggio SQL a seconda del database. Supporta tutti gli operatori standard di LINQ con l'eccezione degli overload che accettano in input l'indice in cui si trova l'elemento. Questa limitazione è presente per ovvi motivi di traduzione in SQL. Oltre agli operatori standard, LINQ to Entities aggiunge un altro operatore (Include).

Entity SQL: Entity SQL è un altro query lenguage come LINQ to Entities. Anche se questo è un po' più difficile di L2E siccome espresso in formato stringa. Questo offre però il vantaggio della creazione dinamica di query. Il motivo per cui esistono due differenti metodi per interrogare il domain model, sta nel fatto che quando si è iniziato a sviluppare Entity Framework LINQ ancora non esisteva e quindi si doveva ricorrere ad un metodo diverso. Allo stato attuale tuttavia L2E è da preferire a Entity SQL.

Object Service: è un punto centrale di EF, uno dei componenti più utilizzati, direttamente o indirettamente, dallo sviluppatore. Ha una moltitudine di compiti, uno dei più importante di questo strato è fornire un accesso alla classe ObjectQuery<T> che rappresenta il punto di entrata per effettuare qualsiasi query. Una volta effettuata la query, sempre l'Object Service si occupa di trasformare i dati dal formato tabellare, ottenuto dallo strato sottostante in classi. Un'altra funzione fondamentale svolta dall'Object Service è quella di tracciare lo stato degli oggetti per capire quali sono stati modificati, inseriti o eliminati così da poter riportare le modifiche sul database nel momento in cui l'utente decide di persisterle

Entity Client Data Provider: questo strato si occupa di convertire le query scritte con L2E o Entity SQL in query SQL. Quindi nel linguaggio capito dal database sottostante. Inoltre, questo strato è responsabile del colloquio fisico con il database poiché gestisce il ciclo di vita della connessione, dei comandi e dei reader.

ADO.Net Data Provider: Questo strato comunica con il database usando ADO.Net standard.

Gli approcci di sviluppo con Entity Framework sono essenzialmente di tre tipi:

- Code First
- Model First
- Database First

Nell'approccio di tipo **code first** tutte le tabelle sono rappresentate da classi che possono essere modificate facilmente a seconda delle esigenze, i cambiamenti possono essere poi distribuiti sul database tramite Code First Migrations.

In **model first** le entità, le relazioni e le varie proprietà vengono create e definite direttamente sul file EDMX. Dopo di che il database viene generato dal modello così formato

Infine, con l'approccio **database first**, vengono prima importate le tabelle del database preesistente. Successivamente vengono generate automaticamente le classi, e per ogni colonna della tabella viene generata un'omonima proprietà nella classe. In base alle informazioni nel database, il designer genera anche le relazioni tra le classi

3.4.3 ASP.NET IDENTITY

Questo framework rende più facile integrare i dati dell'utente all'interno del nostro sito web ed è basato sul framework OWIN (Open Web Server Interface for .NET), un middleware che definisce un layer standard tra applicazione e web server.

Nasce dal progetto ASP.NET Membership per poterne correggere e risolvere alcuni limiti, come:

- Schema e DB erano disegnati per SQL-Server e non potevano essere cambiati;
- Le informazioni aggiunte manualmente ai profili venivano inserite in tabelle esterne alla memberschip e risultava molto difficile accedere a questi dati attraverso il codice;
- LogIn e LogOut non utilizzavano OWIN e non era supportato alcun provider esterno.

La versione più simile ad Identity era “Universal Providers” tuttavia presentava ancora alcune mancanze.

ASP.NET Identity può essere usato con tutti i framework ASP.NET (MVC, Web.Api, Web form, ecc), può essere utilizzato inoltre in applicazioni per Windows phone o Windows 8/10. Attraverso Entity Framework code first risulta molto semplice aggiungere proprietà alle informazioni del profilo di un utente.

Un altro problema risolto rispetto a membership è la possibilità di cambiare lo schema del database, modificandone le tabelle e le relazioni. È stato aggiunto un metodo più semplice anche per la gestione e la creazione dei ruoli.

Questo è un framework basato sulle claims, è quindi molto più semplice accedere ed arricchire le informazioni legate all’utente. Infine, essendo adesso basato su OWIN non è più necessaria la dipendenza da un System.Web.

ASP.NET Identity possiede un design modulare, composto da molte semplici parti che lo rendono facilmente adattabile ad una notevole varietà di scenari. Identity è un prodotto progettato per essere future-ready ed in grado di concorrere con l'elevata longevità di Membership API, suo predecessore.

I componenti di ASP.NET Identity sono raccolti in tre pacchetti ottenibili da NuGet, ciascuno avente specifiche responsabilità.

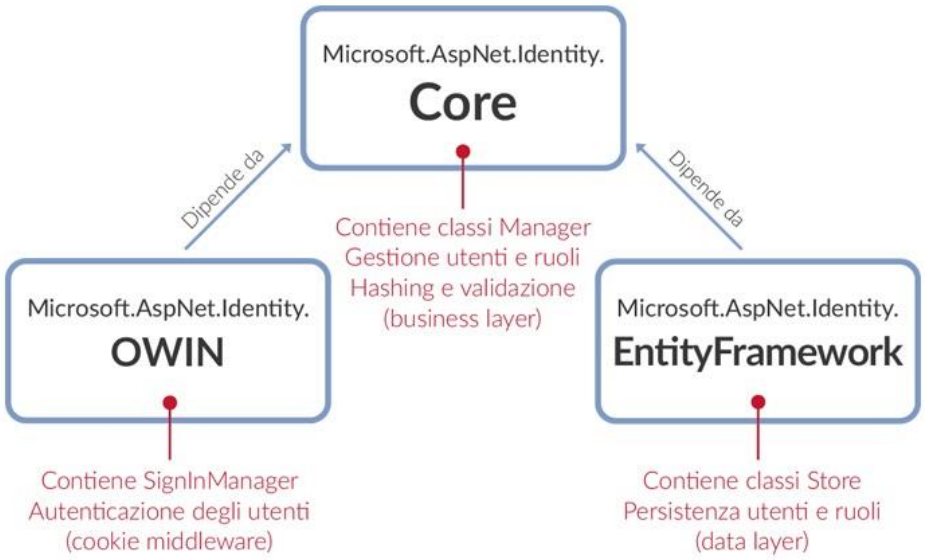


Fig. 7 – componenti Asp.Net Identity

Nel pacchetto Core spicca la classe UserManager, principale API di gestione ad alto livello degli utenti. Sebbene possa essere usato individualmente, sarà certamente utile anche del pacchetto OWIN che si occupa dell'interazione con il mondo HTTP, ovvero di verificare le credenziali. Il pacchetto EntityFramework permetterà invece di persistere gli account con una della tecnologie database supportate.

Già gli antenati di ASP.NET Identity (FormsAuthentication e Membership API) rappresentavano una soluzione immediata e standardizzata di autenticare e gestire gli utenti. Sono trascorsi molti anni e oggi uno sviluppatore è chiamato ad affrontare scenari molto più complessi e mutevoli. Spesso, le applicazioni web si trovano ad operare in contesti sensibili, in cui la sicurezza è un requisito principe e l'account va protetto anche in caso di furto di password.

Si vede ora una rapida carrellata di quali sono le opportunità offerte da ASP.NET Identity 2 in ambito di sicurezza.

Two-factor authentication

La two-factor authentication è volta a rendere più robusta la procedura di login e a limitare l'accesso non autorizzato all'account in caso di furto di password. L'utente non deve solo dar prova di conoscere le sue credenziali di accesso, ma anche di possedere un telefono cellulare o una casella email a cui la nostra applicazione recapiterà un codice di verifica. Possiamo configurare i servizi di recapito SmsService ed EmailService dal file App_Start/IdentityConfig.cs, dove si trova il factory method Create del nostro UserManager.

All'interno del progetto è stato scelto tuttavia di non supportare la two factor authentication. Si è valutato che i dati inseriti non avessero un livello di sensibilità tale da richiedere questo tipo di verifica.

Account lock

Come ulteriore meccanismo di protezione dagli attacchi, si può predisporre un blocco automatico dell'account dopo un certo numero di errori nell'inserimento di password o di codici di verifica della two-factor authentication. Dal file App_Start/IdentityConfig.cs, è possibile configurare la politica di blocco per il nostro ApplicationUserManager, indicando la durata del blocco e il numero di tentativi massimi.

Anche in questo caso l'implementazione all'interno del progetto non è stata prevista.

Conferma account

Uno scenario tipico delle registrazioni è quello del double opt-in, ovvero la verifica dell'indirizzo e-mail tramite click su link di attivazione.

L'invio di una mail per la conferma del indirizzo inserito durate la registrazione invece è stato aggiunto al progetto.

Utenti e ruoli interrogabili con LINQ

Lo UserManager espone le proprietà Users e Roles che, essendo di tipo IQueryable, consentono di interrogare gli utenti e i ruoli con espressioni LINQ arbitrarie.

```
var utentiConTelefono = UserManager.Users.Where(
  user => !string.IsNullOrEmpty(user.PhoneNumber) &&
  user.PhoneNumberConfirmed);
```

Validità della password

Tra le classi helper del pacchetto Core troviamo il PasswordValidator che incapsula la logica di validazione delle password scelte dagli utenti. Dalle sue proprietà, configuriamo i nostri criteri di lunghezza minima e di assortimento di simboli e caratteri alfanumerici.

Va infine specificata una cosa: alcune implementazioni standard di ASP.NET Identity 2 prevedono già l'integrazione di alcuni sistemi di sicurezza. Perciò le tabelle come AspNetUsers presenteranno delle colonne per la two-factor authentication e l'autenticazione tramite cookie oppure in AccountController saranno esposti metodi per l'integrazione con i social network e altre cose di questo genere.

È stato deciso di mantenere queste integrazioni, o quanto meno la maggior parte di esse, siccome in un futuro sviluppo si potrebbe decidere di integrare anche queste funzionalità. Il codice quindi è stato costruito in modo da non entrare in conflitto con queste parti qualora si decidesse di integrarle nella soluzione.

3.4.4 OAuth

OAuth non è un software, ma un protocollo che permette di accedere o rendere disponibili delle API in sicurezza con un metodo standard.

Grazie a questo protocollo un utente può accedere alle sue informazioni su di un servizio (detto service provider) da un applicazione o un altro servizio, detto consumer, senza condividere la propria identità.

L'autenticazione al service provider viene mantenuta tramite un token che garantisce al consumer l'accesso ai dati dell'utente. Il token ha in genere un limite temporale dopo il quale deve essere richiesto nuovamente.

Analizziamo le operazioni da effettuare con OAuth per ottenere l'autenticazione.

1. Il consumer invia al service provider una richiesta per il request code.
2. Il service provider risponde al consumer con un request code
3. Il consumer ridirige l'utente ad una pagina di autenticazione
4. L'utente effettua l'autenticazione sul service provider
5. Il service provider conferma l'avvenuta autenticazione
6. L'avvenuta autenticazione viene inoltrata al consumer
7. Il consumer richiede il token di accesso al service provider
8. Il service provider genera un token e lo invia in risposta al consumer
9. Con il token il consumer è in grado di accedere alle risorse del service provider
10. Il service provider soddisfa le richieste del consumer solo se il token è valido

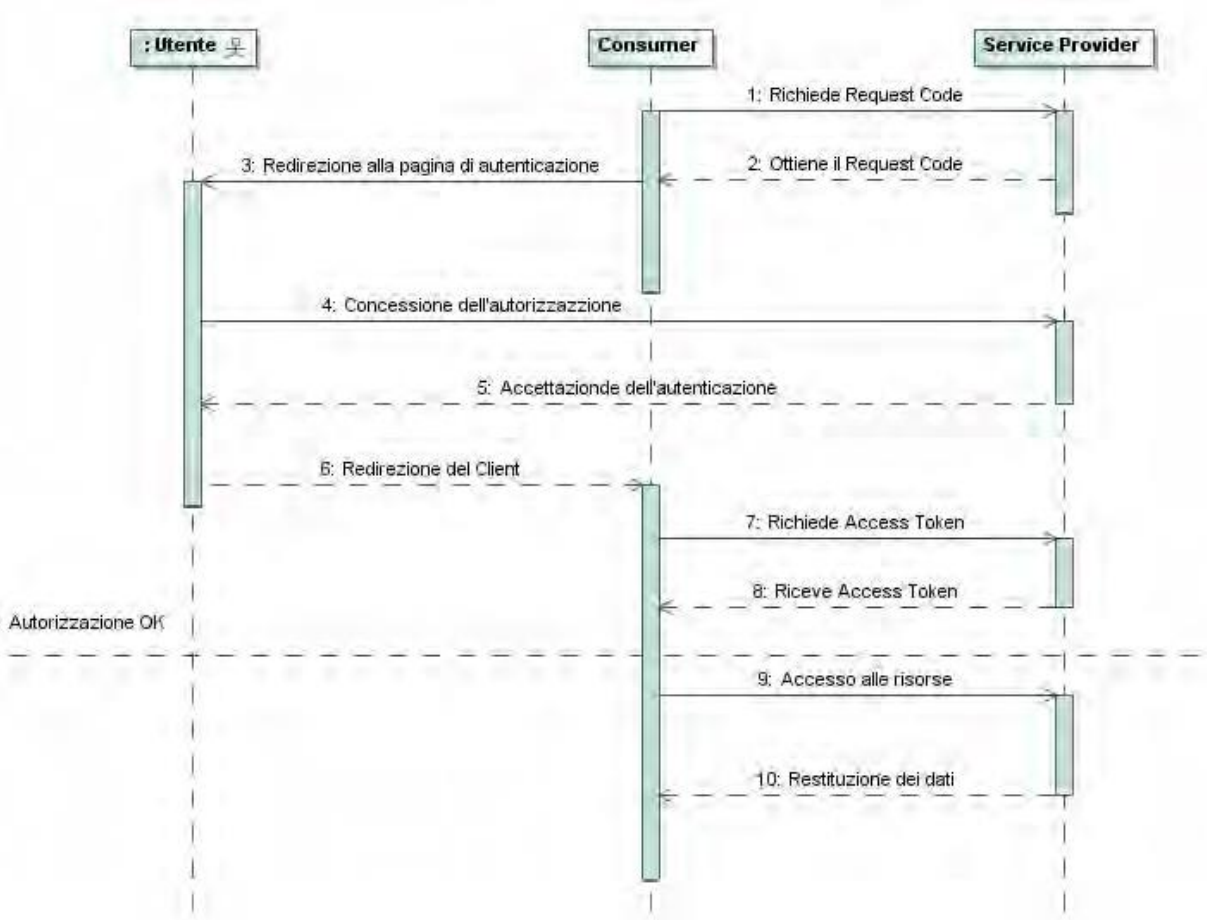


Fig. 8 – Flusso dei dati in una autenticazione con OAuth

L'access token da solo è sufficiente a garantire l'accesso alle risorse ma per garantirne una maggiore sicurezza è necessario che la connessione utilizzi lo schema HTTPS in modo da evitare tentativi di sniffing.

Non è sempre detto che un token garantisca l’accesso alla totalità dell’applicazione. Tramite la definizione di ruoli all’interno del database è infatti possibile limitare, o nascondere, alcune risorse ad una certa tipologia di utenti.

3.4.5 JSON WEB TOKEN

Si tratta di uno standard che viene usato per "regolare" le richieste tra due parti. JSON Web Token è un security token che agisce come un container per le claims degli user. Come si può intuire dal nome, in JWT le claim sono codificate utilizzando JSON, questo rende più facile il loro utilizzo; specialmente in applicazioni basate su JavaScript.

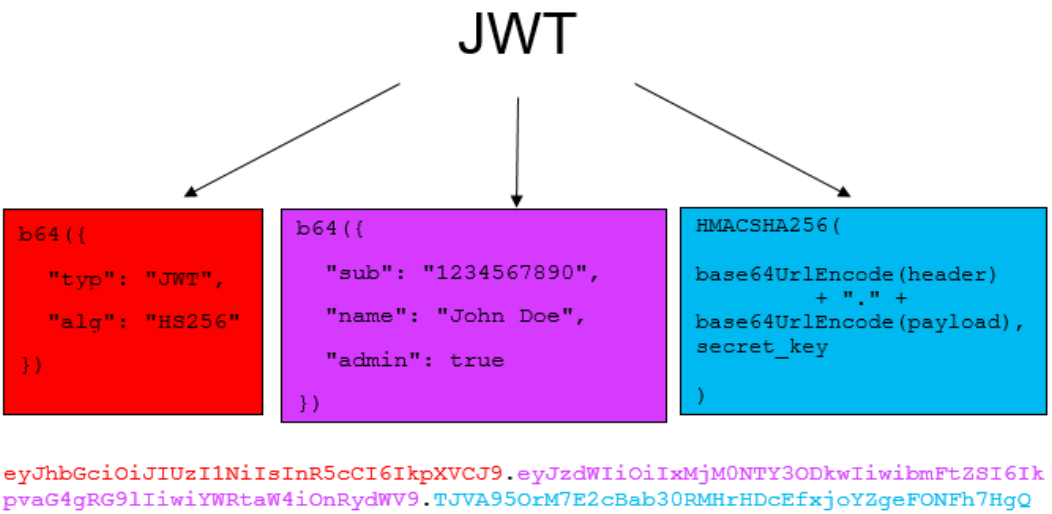
Un JSON Web Token può essere “firmato” seguendo le specifiche di JSON Web Signature (JWS), allo stesso modo può essere criptato attraverso le specifiche del JSON Web Encryption (JWE). In questo progetto però non è stato usato nessun tipo di cifratura, siccome all’interno del JWT non verrà incluso alcun tipo di dato sensibile.

Sostanzialmente un JWT può essere suddiviso in tre parti:

- **Header**: l'intestazione contiene delle informazioni sul token stesso e su come è stato criptato;
- **Payload**: il corpo vero e proprio che contiene dei dati "variabili" in base al contesto;
- **Signature**: una firma che contiene un hash dell'header, del payload ed un "secret" conosciuto solo dai server dell'applicazione;

Le prime due parti, l'header ed il payload, vengono codificate nel formato Base64.

Fig. 9 – Struttura di un token JWT



Header

Come si può notare anche dalla figura sopra l'header contiene sempre le due voci "typ" e "alg". Il primo ha come valore sempre "JWT", mentre il nodo "alg" contiene il nome dell'algoritmo usato per il token. Nell'immagine sopra è stato usato l'algoritmo "HMAC-SHA256".

Playload

La parte di payload è anch'essa un oggetto JSON, il quale contiene tutte le informazioni che vogliamo trasmettere (generalmente riguardanti l'utente)

Signature

La signature è un semplice hash calcolato da un algoritmo descritto nell'header (HMAC-SHA256 in questo caso), usando una parola chiave nota solo al server. Le stringa che viene "hashata" è la semplice concatenazione delle versioni Base64 di header e payload, divise da un punto.

3.5 IMPLEMENTAZIONE SERVER

Durante la fase di implementazione, effettuata con Microsoft Visual Studio Ultimate 2013 e l'ultima versione di ASP.NET Identity, si è scelto di apportare qualche cambiamento al database precedentemente progettato. Le modifiche riguardano esclusivamente la tabella "clienti". Si è infatti deciso di utilizzare le tabelle di default che il sistema genera se, durante la creazione del nuovo web project, viene selezionata l'opzione "Individual User Accounts" come tipo di autenticazione.

Sono perciò state introdotte quattro nuove tabelle al posto di quella che, in fase di progettazione, era stata indicata come "clienti". Tre fanno parte delle tabelle create automaticamente dal sistema (AspNetUsers, AspNetUsersRoles, AspNetRoles) mentre la quarta, denominata "UserProfileInfoes", è usata per le informazioni personali dell'utente e per mantenere le relazioni che possedeva la tabella "clienti".

Queste scelte sono state operate per facilitare la gestione dei ruoli all'interno del server e perché queste tabelle forniscono degli spunti interessanti per sviluppi futuri. Mentre la tabella "UserProfileInfo" è stata creata separata per non intaccare la struttura della tabella originale "AspNetUsers" e per non caricarla ulteriormente sia attributi, ma soprattutto di relazioni.

In seguito verranno specificati i seguenti punti salienti per quanto riguarda l'implementazione effettuata per il server:

- Autenticazione e autorizzazione;
- Gestione dei ruoli;
- Validazione mail;
- Controller

Ma prima ecco lo schema ottenuto dagli aggiustamenti descritti sopra

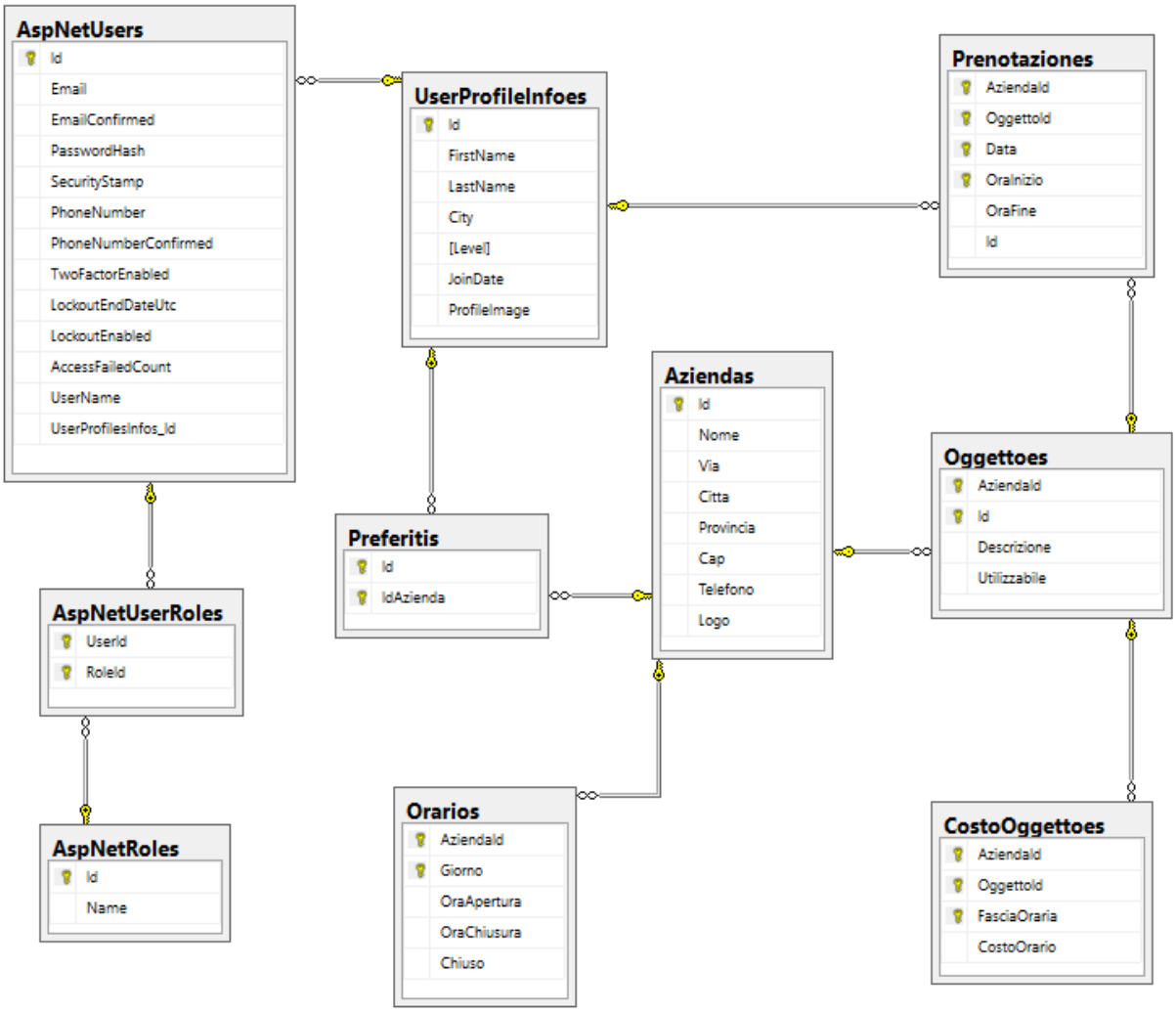


Fig. 10 – Schema del database con revisioni finali post progettazione

3.5.1 AUTENTICAZIONE E AUTORIZZAZIONE

È stata implementata un tipo di autenticazione token-based. Un token verrà quindi utilizzato per autenticare l'utente ogni volta che giungerà una richiesta al server.

Tendenzialmente lo schema base per questo tipo di autenticazione sarà il seguente:

1. Lo user inserisce uno username e una password nel form di login e clicca Log In
2. Dopo che la richiesta è stata inviata, il backend cercherà di validare lo user effettuando una query sul database. Se la ricerca produce un risultato sarà generato un nuovo JWT access token ed

inviato nuovamente al client per essere poi memorizzato localmente.

3. Nell'header di ogni richiesta successivamente effettuata dovrà essere incluso anche il token per poter accedere alle aree protette dell'applicazione
4. Se il token incluso nella richiesta è valido il server lascerà accedere l'utente all'area protetta, rispondendo con del JSON o del XML. Altrimenti fornirà un messaggio di errore.

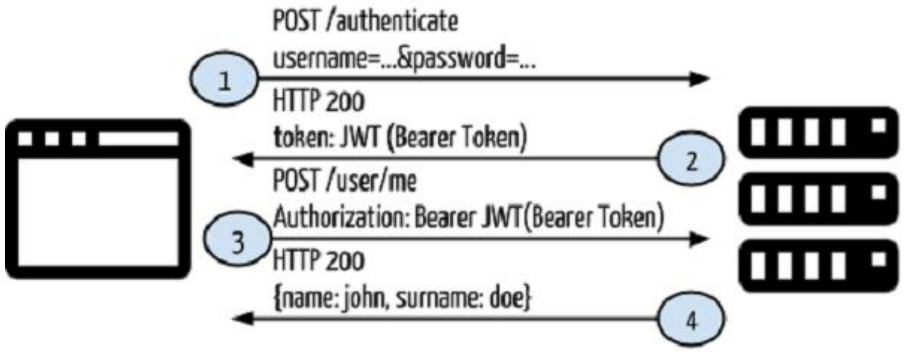


Fig. 11 – Sistema di autenticazione token-based

Il flusso di credenziali dello user è gestito dalla classe "CustomOAuthProvider" questa classe eredita da "OAuthAuthorizationServerProvider" ed effettua un override sui seguenti due metodi:

ValidateClientAuthentication una richiesta viene considerata sempre valida siccome in questo caso il client (l'applicazione Android) è un client fidato. Non si ha quindi il bisogno di validarlo.

GrantResourceOwnerCredentials si occupa di ricevere username e password dalla richiesta e cercare di validarli affidandosi ad ASP.NET 2.1 Identity. Se le credenziali sono valide e l'email è confermata si procederà alla costruzione di un identity allo user loggato. Questa conterrà tutti i ruoli e le eventuali claim dello user autenticato.

Di seguito l'immagine del metodo "GrantResourceOwnerCredentials"

```
public override async Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)
{

    const string allowedOrigin = "*";

    context.OwinContext.Response.Headers.Add("Access-Control-Allow-Origin", new[] { allowedOrigin });

    var userManager = context.OwinContext.GetUserManager<ApplicationUserManager>();

    ApplicationUser user = await userManager.FindAsync(context.UserName, context.Password);

    if (user == null)
    {
        context.SetError("invalid_grant", "The user name or password is incorrect.");
        return;
    }

    if (!user.EmailConfirmed)
    {
        context.SetError("invalid_grant", "User did not confirm email.");
        return;
    }

    ClaimsIdentity oAuthIdentity = await user.GenerateUserIdentityAsync(userManager, "JWT");

    var ticket = new AuthenticationTicket(oAuthIdentity, null);

    context.Validated(ticket);

}
```

Il metodo “GenerateUserIdentityAsync” si trova dentro la classe “ApplicationUser”, recupera dal database l'identità dell'utente autenticato e restituisce un oggetto di tipo "ClaimsIdentity".

```
public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager, string
authenticationType)
{
    var userIdentity = await manager.CreateIdentityAsync(this, authenticationType);

    //aggiungo claim con i dati del cliente, per farle poi risultare nel jwt
    Claim idClaim = new Claim("Id", UserProfilesInfos.Id.ToString(CultureInfo.InvariantCulture));
    userIdentity.AddClaim(idClaim);

    Claim firstNameClaim = new Claim("First_Name", UserProfilesInfos.FirstName);
    userIdentity.AddClaim(firstNameClaim);

    Claim lastNameClaim = new Claim("Last_Name", UserProfilesInfos.LastName);
    userIdentity.AddClaim(lastNameClaim);

    Claim cityClaim = new Claim("City", UserProfilesInfos.City);
    userIdentity.AddClaim(cityClaim);

    // Add custom user claims here
    return userIdentity;
}
```

Alla fine di “GrantResourceOwnerCredentials” verrà chiamato “context.Validated(ticket)”, che trasferirà l’identità così creata ad un OAuth 2.0 bearer access token.

Fino a questo momento il server non è stato ancora configurato per l'utilizzo del OAuth 2.0 Authentication Workflow. Per fare ciò nella classe "Startup" è stato creato un metodo chiamato "ConfigureOAuthTokenGeneration".

Questo è un po' il fulcro del applicazione per quanto riguarda l'autenticazione di un utente.

```
private void ConfigureOAuthTokenGeneration(IAppBuilder app)
{
    // Configure the db context and user manager to use a single instance per request
    app.CreatePerOwinContext(ApplicationDbContext.Create);
    app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);
    //Ora una singola istanza di classe "ApplicationRoleManager"
    //sarà disponibile per ogni richiesta, useremo questa istanza con diversi controller
    app.CreatePerOwinContext<ApplicationRoleManager>(ApplicationRoleManager.Create);

    var OAuthServerOptions = new OAuthAuthorizationServerOptions()
    {
        AllowInsecureHttp = true,
        //il percorso per generare il JWT sarà http://localhost:16393/Token
        TokenEndpointPath = new PathString("/Token"),
        AccessTokenExpireTimeSpan = TimeSpan.FromHours(6),
        Provider = new CustomOAuthProvider(),

        //Abbiamo inoltre specificato che il token andrà generato utilizzando il
        //formato JWT con la classe "CustomJwtFormat". La classe genererà un token
        //JWT invece dello standard DPAPI
        AccessTokenFormat = new CustomJwtFormat("http://localhost:16393")
    };

    // OAuth 2.0 Bearer Access Token Generation
    app.UseOAuthAuthorizationServer(OAuthServerOptions);
}
```

Si è specificato il percorso da contattare per generare il token JWT (http://localhost:16393/Token), la sua scadenza (6 ore), che criteri seguire per validarlo (CustomOAuthProvider) ed infine come costruire quest'ultimo. La classe "CustomJwtFormat" conterrà infatti le linee guida per la creazione di questo particolare token.

Ora è possibile proteggere gli end points, di un qualsiasi controller, da accessi di utenti non registrati semplicemente apponendogli l'attributo `[Authorize].`

A questo punto però la API non è ancora in grado di capire i JWT emessi, andrà perciò aggiunta una classe che avrà il compito di gestire i token presenti nelle richieste che sopraggiungono.

È stata quindi aggiunto un nuovo metodo alla classe "Startup": "ConfigureOAuthTokenConsumption"

```
private void ConfigureOAuthTokenConsumption(IAppBuilder app)
{
    var issuer = "http://localhost:16393";
    string audienceId = ConfigurationManager.AppSettings["as:AudienceId"];
    byte[] audienceSecret =
TextEncodings.Base64Url.Decode(ConfigurationManager.AppSettings["as:AudienceSecret"]);

    //I controller con un attributo [Authorize] saranno validati con un JWT
    app.UseJwtBearerAuthentication(
        new JwtBearerAuthenticationOptions
        {
            AuthenticationMode = AuthenticationMode.Active,
            AllowedAudiences = new[] { audienceId },
            IssuerSecurityTokenProviders = new IIssuerSecurityTokenProvider[]
            {
                new SymmetricKeyIssuerSecurityTokenProvider(issuer, audienceSecret)
            }
        });
}
```

Fornendo tali valori al middleware "JwtBearerAuthentication", la nostra API sarà in grado di consumare solo token JWT emessi dal nostro server di autorizzazione di fiducia, eventuali altri token JWT da qualsiasi altro server di autorizzazione saranno respinti.

3.5.2 ROUTING

Il routing, cioè l'indirizzamento delle richieste per uno specifico URL verso il controller preposto, viene effettuato sfruttando l'architettura messa a disposizione da ASP.NET Web API 2.

La prima release di Web API usava un tipo di routing chiamata “convention-based”, in questa tipologia si definivano una o più route template, che erano sostanzialmente delle stringhe parametrizzate. Quando il framework riceveva una richiesta andava a cercare un matching con i template precedentemente definiti. Sfortunatamente questo tipo di routing non supportava certi tipi di URI molto frequenti nelle API RESTful. Per esempio:

```
/customers/1/orders
```

questo tipo di URI era difficile da creare usando un tipo di routing “convention-based”. È però possibile ottenerla tramite un tipo di routing diverso, chiamato “attribute routing”. Può infatti essere ottenuto aggiungendo sopra alla action del controller l’attributo `[Route("customers/{customerId}/orders")]`.

Il codice responsabile per parte di routing si trova dentro la classe "WebApiConfig" dentro la cartella "App_Start". Qui, rispetto alla versione precedente di Web Api, è stata inserita la riga `config.MapHttpAttributeRoutes()` per attivare l'attribute routing.

Attribute Routing può essere combinato con la versione precedente (convention-based). Si nota infatti che sotto all'attivazione dell'attribute routing, si trova una chiamata al metodo `MapHttpRoute` per definire una route di tipo conversion-based.

```
config.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

Il modo convenzionale con il quale Web Api effettua il routing per le richieste in arrivo è quello di inviarle al controller specificato nell'URI. Successivamente viene cercata una corrispondenza tra il metodo specificato nella richiesta http e un action method il cui nome inizia con quel metodo.

Un file simile a "WebApiConfig" lo si trova sempre all'interno della cartella "App_Start" denominato "RouteConfig". Esso è lo stesso che si può trovare all'interno di un progetto MVC ed infatti serve per effettuare il route sul framework MVC.

3.5.3 GESTIONE DEI RUOLI

È stato visto come, grazie all'attributo `[Authorize]` sia possibile proteggere un end-point da accessi di utenti non registrati. Tuttavia questo metodo presenta alcuni problemi, siccome permette ad un qualsiasi user autenticato di eseguire ogni tipo di azione; anche operazioni sensibili, come la cancellazione o l'inserimento di nuove informazioni.

Per far fronte a questo problema si è deciso di sfruttare il supporto che ASP.NET Identity 2.1 offre per la gestione dei ruoli; possiamo così

controllare più facilmente le azioni che un utente può fare in base al ruolo che assume.

Come si può notare nel metodo visto in precedenza "ConfigureOAuthTokenGeneration" è stata inserita una riga di codice per poter creare una singola istanza della classe "ApplicationRoleManager" per ogni richiesta. "ApplicationRoleManager" si avvale del middleware OWIN per creare la richiesta.

Con questi accorgimenti è ora possibile limitare l'accesso ad un metodo esposto da un controller solamente a user autenticati, che appartengono ad un determinato ruolo. È possibile grazie all'aggiunta di `(Roles = "Nome_Ruolo")` all'attributo `[Authorize];` ottenendo, per esempio, `[Authorize(Roles = "Admin")]`

3.5.4 VALIDAZIONE EMAIL

La tabella di ASP.NET Identity 2.1 (AspNetUSers) viene creata di default con l'attributo di tipo boolean "EmailConfirmed", questa colonna è usata per memorizzare se l'email fornita dallo user è valida e appartiene ad esso. In altre parole si vuole verificare se l'utente ha accesso alla mail che ha fornito, e che non stia cercando di impersonare qualcun altro. Quindi il server non dovrà permettere agli utenti senza una mail valida, di accedere a parti protette dell'applicazione.

Quello che si è voluto implementare è un sistema che, dopo la registrazione dell'utente, spedisca una mail all'indirizzo specificato nel momento della registrazione. Questa email includerà un link per l'attivazione, utilizzabile solo da quell'utente e con un tempo di validità limitato. Una volta che l'utente avrà aperto la mail e cliccato sul link di attivazione (e se il codice è ancora valido) il campo "EmailConfirmed" sopra citato sarà settato a "true". Confermando così il fatto che la mail appartiene all'utente che sta cercando di effettuare la registrazione.

Per implementare quanto detto finora è necessario aggiungere un servizio responsabile di inviare le mail. Si è scelto come servizio Sand Grid (https://sendgrid.com/) il quale fornisce la possibilità di inviare fino a 400 email al giorno con un account gratuito.

Tramite NuGet si è installato il package contenente le API di Send Grid.

È stato poi necessario modellare una classe con il compito di creare, ed inviare la mail all'indirizzo fornito dell'utente. Questa classe è stata chiamata "EmailService".

Una volta impostato "EmailService" dovrà essere integrato con il sistema di sicurezza del server. Ciò è stato possibile semplicemente aggiungendo al metodo "Create", all'interno della classe "IdentityConfig", il riferimento alla classe precedentemente creata, e settando la validità del codice (token), impostata a 6 ore.

Si hanno ora tutti i mezzi per inviare una email, dopo l'avvenuta creazione di un account. Si è andato perciò a modificare il codice esistente del metodo “Register” nel controller “AccountController”. È stato inserito: prima il codice per generare il token che sarà valido per le prossime 6:00 ore e legato ad un solo user; poi si è generato il link di attivazione da includere nel body dell'email.

Abbiamo, adesso, bisogno di costruire un percorso nella nostra API che riceva la richiesta quando l'utente fa clic sul link di attivazione ed emetta una richiesta HTTP GET, per farlo c’è bisogno di implementare un nuovo metodo nel controller “AccountController”. Questo controllerà che i parametri passatigli (userId e codice) non siano nulli, dopo di che si affiderà al metodo “ConfirmEmailAsync” per convalidarne i valori.

Quindi se l’Id utente non è legato a questo codice o anche se il codice è scaduto, la richiesta fallirà. Se invece tutto va bene questo metodo aggiornerà il campo “EmailConfirmed” nella tabella "AspNetUsers" e lo imposterà a "True".

4 FIND TO PLAY LATO CLIENT

In questo capitolo verrà analizzata la parte progettuale e implementativa dell'applicazione lato client per Android.

L'obiettivo fondamentale di questa tesi è quello di creare un software in grado di aiutare l'utente a trovare le informazioni riguardanti i centri sportivi più vicini, controllarne le prenotazioni e, nel caso, effettuarne una. Cerca quindi di semplificare in parte l'organizzazione di un evento sportivo o di una partita tra amici che comprenda la prenotazione di un campo per un certo numero di ore.

Inoltre find to play prova anche ad essere un mezzo con il quale i centri sportivi possono acquistare più visibilità, fornendo al possibile cliente tutte le informazioni necessarie senza dover aprire un sito o avere una pagina su un qualche social.

Prima di procedere con la stesura del codice, è necessario effettuare analisi sulle funzionalità che l'app deve fornire.

4.1 ANALISI DELLE FUNZIONALITÀ

L'applicazione deve permettere la registrazione di nuovi utenti nonché l'inserimento dei dati basilari. Un utente registrato deve poter effettuare una prenotazione, mentre uno user anonimo (non registrato o che non ha effettuato il log in) deve poter effettuare ricerche, vedere tutte le informazioni riguardanti un centro sportivo e le prenotazioni effettuate. Un utente registrato può aggiungere un centro sportivo ad un elenco di preferiti. Deve poter inoltre accedere ad una sezione dell'applicazione dove è possibile visualizzare tutte le sue informazioni inserite al momento della registrazione. Deve infine poter modificare, o cancellare, una prenotazione (entro un certo tempo prima della data prenotata)

4.2 SCELTA DI SOFTWARE E TECNOLOGIE

Per lo sviluppo del client è stato usato, come già detto in precedenza, Android. Decisione presa in base alla diffusione dei sistemi e a disponibilità tecnologiche. Sul sistema sono state poi utilizzate diverse librerie. Come Google Play Services o Volley, messe a disposizione da Google stessa. Sono state integrate al progetto anche alcune librerie sviluppate da privati e condivise, in modo gratuito, su gitHub; come: Android Week View o Dynamicbox. Va infine specificato che lo sviluppo dell'applicazione è stato fatto cercando di rispettare quelle che sono le linee guida del material design di Google.

4.2.1 GOOGLE PLAY SERVICES

Con questa libreria l'app può usufruire delle ultime features sviluppate dalla "grande G". Sono accessibili servizi come l'integrazione con Maps, Google+ ed altri. L'APK di Google Play services gira come un processo in background in Android. L'utente interagisce con il servizio in background attraverso le librerie a disposizione del client.

La libreria è distribuita tramite lo store di Android (Google Play) in modo che il processo di aggiornamento sia molto più semplificato. Quasi senza che l'utente se ne accorga.

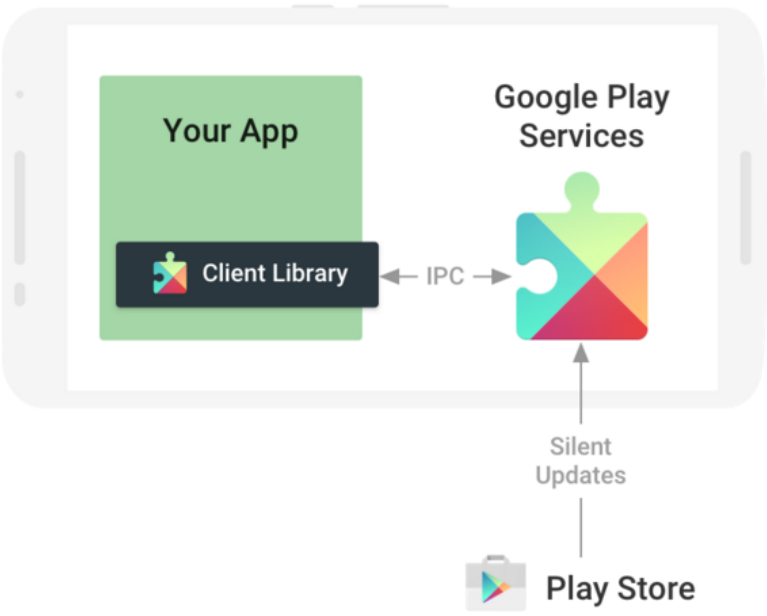


Fig. 12 – Architettura utilizzata per i Google Play Services

4.2.2 VOLLEY

La maggior parte delle applicazioni che, come Find to Play, prevedono un interazione fra gli utenti più o meno diretta, si appoggiano su un set di API REST fornite dal server con cui i terminali degli utenti dialogano.

Per lo sviluppatore la difficoltà maggiore nello sviluppo di questo pattern non consiste tanto nella difficoltà del lavoro da svolgere quanto nella ripetitività del codice da scrivere. Ci troviamo di fronte a quello che solitamente viene chiamato in informatica codice boilerplate.

Il programmatore si trova infatti a distrarsi dal problema principale cui sta lavorando per concentrarsi su attività di pura routine, necessarie all'accesso alla Rete.

Volley è una libreria in grado di svolgere in autonomia tutte le operazioni preliminari necessarie, lasciando al programmatore il compito di personalizzare il dialogo con i server remoti contattati. È stata presentata durante il Google I/O del 2013 e, stando alle dichiarazioni fatte durante l'evento, questa libreria è utilizzata in gran parte delle applicazioni di punta di Google stessa.

La libreria è stata definita da Google stessa in questo modo:

"*Volley it's an HTTP library that makes networking for Android apps easier and most importantly, faste*r"

Di seguito si vedranno alcune delle principali caratteristiche della libreria:

- **scheduling automatico (con gestione della priorità) delle richieste di rete**. Volley infatti mette a disposizione una coda e rende trasparente all'utente la gestione della priorità. Lo sviluppatore deve solo aggiungere la request alla coda e fornire una callback. Volley pensa al resto.

- **gestione trasparente della concorrenza**. Tutti i metodi di Volley sono thread-safe per cui è possibile per lo sviluppatore gestire l'aggiunta e rimozione delle request con la garanzia che lo stato degli oggetti sarà protetto dalle modifiche concorrenti. Le richieste a loro volta vengono gestite in parallelo grazie a diversi thread concorrenti.
- **caching trasparente in memoria o su SD.** Lo sviluppatore non deve preoccuparsi nemmeno del caching delle richieste e delle risorse. Basta dichiarare una determinata risorsa REST come cachable e configurarne l'expiration e Volley gestirà automaticamente il caching in modo compliant al protocollo HTTP.
- **possibilità di cancellare una request**. Questa feature è determinante per gestire correttamente la relazione fra il lifecycle delle activity Android e la coda di richieste REST.
- **framework di logging e debugging molto potente**. Sviluppare un client REST, proprio per la natura asincrona delle risposte rispetto alle richieste comporta spesso una difficoltà maggiore nelle fasi di debug. Volley offre una serie di primitive avanzate che lo supportano in queste fasi.

Lavorare con Volley richiede di prendere confidenza con almeno due concetti fondamentali:

- la Request: oggetto che rappresenta la descrizione dell'attività che vogliamo svolgere in Rete comprensiva di indirizzo da contattare, metodo HTTP da utilizzare, codice di reazione sia in caso di successo che di fallimento;
- la RequestQueue, la coda delle richieste: è la struttura dati in cui vengono inserite le richieste preparate dal programmatore. Volley si occuperà di gestirne i tempi di esecuzione in base alle proprie politiche.

Preparare richieste ed inserirle in coda: nient'altro. Queste sono le principali operazioni che il programmatore dovrà essere in grado di svolgere.

4.2.3 LIBRERIE ESTERNE

All'interno del progetto si è scelto di utilizzare alcune librerie gratuite sviluppate da terzi. Si vede ora una breve descrizione delle library di maggior sostanza.

Android Week View

Questa libreria consente di integrare agevolmente, all'interno dell'applicazione, un calendario. A differenza, ad esempio, di quello standard messo a disposizione da Google; Android Week View consente la visualizzazione di uno, tre o sette giorni nella stessa schermata. Non c'è ancora una implementazione di una vista per l'intero mese. Per di più offre anche una visualizzazione agevole degli eventi organizzandoli per orario, quasi come in un'agenda.

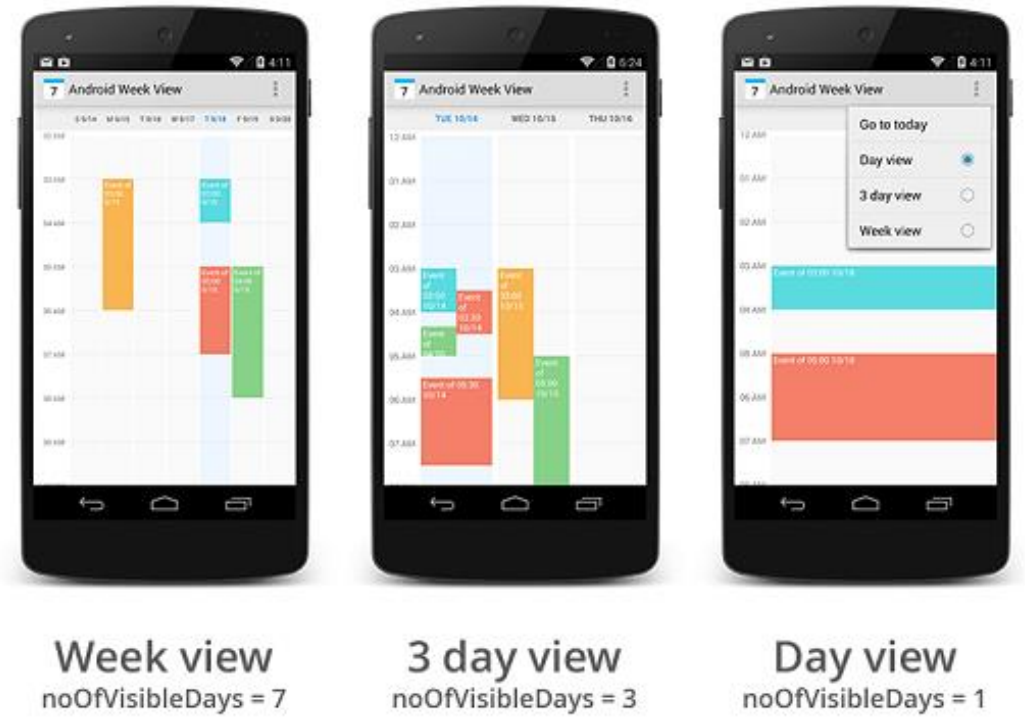


Fig. 13 – Screenshot dei tre tipi di view che offre la libreria

Nimbus Jose + Jwt

Sviluppata da connect2id; è, stando al loro sito web, una delle più popolari e robuste librerie java per JSON Web Tokens (JWT). È una libreria open source e supporta tutti i metodi standard la criptazione (JWE) e la firma (JWS). La libreria richiede Java 7+ come requisiti minimi

La libreria è molto vasta, permette di creare un token (JWT/JWS/JWE), di effettuarne il parsing, di creare RSA, EC and symmetric JSON Web Keys (JWKs). Tuttavia, per lo sviluppo dell'applicazione, sono state usate solo le funzionalità di lettura e di estrapolazione dei dati offerte da questa.

Inizialmente, vista la mole di funzionalità che sarebbero state inutilizzate in questa libreria, si erano cercate soluzioni più semplici e snelle. Ma tutte quelle trovate presentavano qualche problema di integrazione o di funzionamento. Si è pertanto scelto alla fine di integrare ugualmente questa Ninbus Jose+Jwt siccome unica libreria che non presentava alcun tipo di errore.

Material DateTime Picker

Questa libreria cerca di offrire dei picker di data e ora come vengono mostrati nelle Material Design spec.

Per poter ottenere la data o l'orario impostati nei picker andranno implementate le interfacce `OnTimeSetListener` e `OnDateSetListener`.

```
@Override
public void onTimeSet(RadialPickerLayout view, int hourOfDay, int minute) {
  String time = "You picked the following time: "+hourOfDay+"h"+minute;
  timeTextView.setText(time);
}

@Override
public void onDateSet(DatePickerDialog view, int year, int monthOfYear, int dayOfMonth) {
  String date = "You picked the following date: "+dayOfMonth+"/"+(monthOfYear+1)+"/"+year;
  dateTextView.setText(date);
}
```

La libreria offre tutta una serie di controlli molto interessanti da applicare ai due picker. Motivo per cui è stata scelta questa libreria piuttosto di altre o dell'implementazione standard offerta da Android Studio.

Se ne vedono di seguito un paio:

`TimePickerDialog setTitle(String title)`
Mostra un titolo in cima al pickerDialog

`setMinTime(Timepoint time)`
Imposta il minimo tempo valido che può essere selezionato. Il tempo precedente verrà disattivato

`setMaxTime(Timepoint time)`
Imposta il massimo tempo valido che può essere selezionato. Il tempo successivo verrà disattivato

Questi, insieme ad alcuni altri, sono stati utilizzati in fase di implementazione, ma sono solo una piccola parte rispetto a tutti i controlli messi a disposizione dalla libreria.

4.2.4 MATERIAL DESIGN

All'interno dell'applicazione si è tentato, per quanto possibile, di utilizzare lo standard imposto dal material design. Si vede di seguito una breve introduzione a riguardo.

Il material design è il "linguaggio" con cui Google ha declinato la grafica di tutti i suoi prodotti, da Gmail fino sistema operativo mobile Android (da Lollipop in avanti).

Nasce innanzitutto in contrapposizione e allo stesso tempo come sintesi tra il Flat Design o Metro Style di Microsoft e lo Scheumorfismo dei primi iPhone della Apple.

Quindi il material design non è altro che uno stile, un codice, un linguaggio di design. È lo stile con cui Google ha deciso di rinnovare

tutti i suoi prodotti e sotto il quale gestirli tutti quanti allo stesso modo, con gli stessi principi di esteriorità grafica.

È stato presentato il 25 giugno 2014 durante il Google I/O, l'annuale conferenza dell'azienda rivolta a tutti i vari programmatori e sviluppatori di applicazioni e dispositivi con software e sistemi operativi di mamma Google.

Il designer che ha gestito tutto questo processo creativo ha spiegato che:

"Proprio come la carta, il nostro materiale digitale si può espandere o restringere riformandosi in modo intelligente. I materiali hanno superfici fisiche e bordi. Cose come ombre e cuciture forniscono il significato di quello che tocchi."

Nonostante il material design sia stato lanciato relativamente da poco; è sicuramente già un punto di riferimento e una tendenza in tutto l'ambito del web design e del design di interfacce grafiche, il cosiddetto interaction design. Ma il material sta diventando la nuova tendenza anche nell'ambito del graphic design con le sue forme, la sua struttura e i suoi colori.

4.3 SVILUPPO APP

Per lo sviluppo dell'app lato dell'app su Android è stato utilizzato Android Studio.

Il software è un derivato di IntelliJ IDEA, uno dei principali IDE per Java. Viene fornito da Google e include già l'SDK.

Molto comoda è la presenza di Grandle per la gestione delle dipendenze che ci permette di aggiungere librerie semplicemente indicandole nel file apposito.

I test sull'applicazione non sono stati effettuati utilizzando un emulatore poiché quello fornito con l'SDK risulta poco performante e molto esoso in termini di memoria, si è quindi optato per il test dell'app principalmente su due. Il mio telefono personale, un Samsung Galaxy

S5 che monta Android 5.0 (API 21), e un Samsung Galaxy Nexus sul quale gira Android Stock 4.3 (API 18)

Verranno ora descritte le principali componenti dell'applicazione accompagnati da alcuni screenshot della stessa.

4.3.1 SPLASH ACTIVITY

Questo è il punto di partenza dell'app, ogni volta che viene lanciata l'applicazione viene prima visualizzata questa activity.

Fig. 14 – Splash activity screenshot

Appena visibile l'activity va a controllare la memoria interna del dispositivo (le SharedPreferences) per l'expiration time del token. Quest'ultimo ha validità di sei ore; viene quindi controllato che la data letta sia superiore alla data odierna. Nel caso il controllo sia positivo token è ancora valido, si viene dunque dirottati verso la FrontActivity. In caso contrario la splash cercherà di reperire (sempre nelle SharedPreferences) le credenziali dell'utente, ovvero email e password.

Se anche uno solo dei campi cercati risulta nullo allora potrebbe essere una prima installazione oppure l'utente ha effettuato precedentemente un log-out; in questo caso si verrà reindirizzati verso l'activity di log in.

Infine, si vede il caso in cui il token non è più valido, però vengono trovati sia l'email dell'utente sia la password. Attraverso le sue credenziali dunque si cercherà di fare un log-in automatico. Senza che l'utente faccia niente si interroga il server per un token valido da poter utilizzare. Siccome questa è un operazione che richiede certamente più tempo rispetto alle altre verrà attivata una progress bar impostando `setIndeterminate(true)` così che l'utente non pensi ad un mal funzionamento dell'applicazione.

4.3.2 LOG-IN ACTIVITY

Questa activity permette all'utente di eseguire il login o di raggiungere l'activity di registrazione tramite il link posto sotto il bottone "accedi".

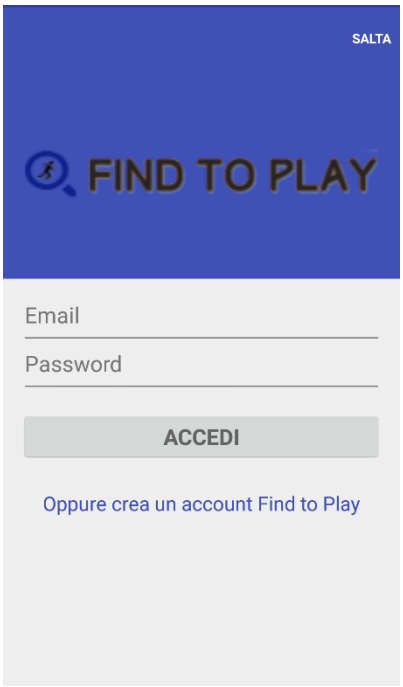


Fig. 15 – Log-in activity screenshot

Si è scelto di non rendere la registrazione vincolante al 100%. Un utente, tramite il tasto "Salta" in alto a destra può cominciare a cercare centri sportivi, vedere le loro informazioni, i campi disponibili e le prenotazioni effettuate. L'unica limitazione che un utente non registrato ha rispetto a un'utente autenticato è quella di non poter aggiungere una nuova prenotazione.

4.3.3 FRONT ACTIVITY

La front activity, come si evince dal nome, è la schermata con il quale l'utente (autenticato oppure no) può iniziare la un centro sportivo.

Fig. 16 – Front activity utente autenticato e non

Come si può notare dai due screenshot sopra l'unica cosa che cambia è che se un utente ha effettuato log-in, in alto a destra, verrà esposto il suo nome. Cliccando su di essa l'utente viene portato sulla pagina dove sono contenute le informazioni di profilo inserite dell'utente.

Se non si è fatto il log-in invece comparirà la scritta "Accedi" che porterà alla schermata di log-in.

4.3.4 PROFILE ACTIVITY

Profile ha lo scopo di permettere la visualizzazione del profilo di un utente. Include inoltre altre funzionalità:

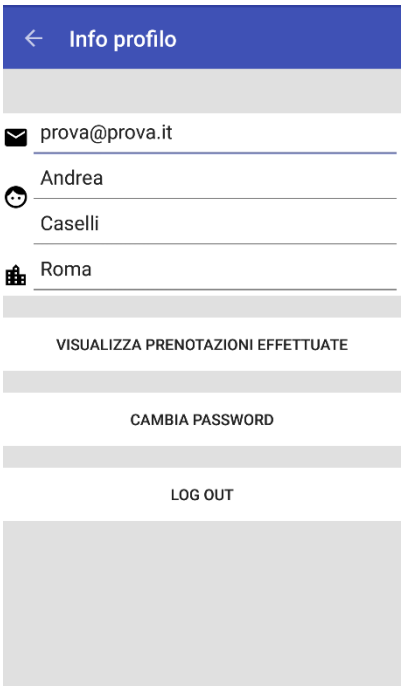


Fig. 17 – Profile activity screenshot

Si nota che sotto alle informazioni fornite dall'utente nel momento della registrazione si può accedere alla view che fa visualizzare tutte le prenotazioni future e passate che l'utente ha eseguito. Così come si può cambiare la password dell'account con una nuova; mentre il terzo ed ultimo bottone permette di fare log-out dall'applicazione. In questo ultimo caso tutti i dati presenti nelle SharedPrefs vengono cancellati e si viene dirottati alla schermata di log-in.

4.3.5 PRENOTATION CLIENTE ACTIVITY

In questa activity un cliente registrato può vedere tutte le prenotazioni fatte. Quelle passate e quelle che ancora dovranno avere luogo.

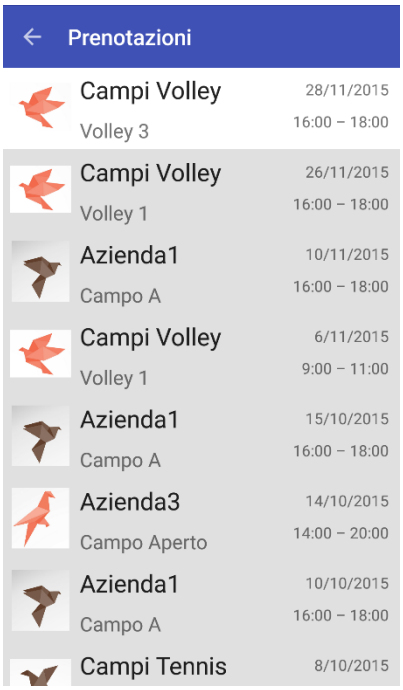


Fig. 18 – screenshot prenotation cliente activity

Come si può vedere dalla figura l'applicazione fa una distinzione tra le prenotazioni già passate e quelle che ancora dovranno realizzarsi. Verrà applicato dunque un colore bianco a queste ultime e un colore grigio chiaro alle altre.

4.3.6 DETTAGLICAMPI ACTIVITY

Al suo interno si trovano due tab; una esporrà le informazioni del centro sportivo che si è cercato e l'altra i campi messi a disposizione per essere prenotati.

Inizialmente erano nate come due activity separate. Dettagli activity e campi activity. Ma poi si è deciso di accorpale in un'unica view siccome era più naturale pensare a queste due come un unico oggetto. Le informazioni sul centro sportivo ed i suoi relativi campi sono due cose strettamente collegate. Si è pensata anche alla user experience, un utente dopo aver visto le informazioni come: via, città, numero di

telefono, e gli orari vuole vedere subito i campi prenotabili. Si è deciso perciò che delle tabs avrebbero pienamente soddisfatto questa necessità.

Fig. 19 – funzionamento della activity dettagliCampo

Come si può vedere la prima tab è chiamata “Dettagli” qui, oltre che le informazioni anticipate pocanzi vengono visualizzate anche una mappa per vedere dov’è situato il posto e un numero di telefono che, una volta toccato, aprirà immediatamente l’applicazione telefono, già impostata per far partire la chiamata. Siccome la tab dettagli è una scrollView, inizialmente operare con il fragment della mappa risultava molto difficile. Si è deciso quindi di togliere la precedenza di movimento alla scrollView nel caso l’utente cercasse di interagire con la mappa. Inoltre, cliccando il segno rosso sulla mappa, compariranno i controlli che permettono di settare il navigatore per quella posizione.

La seconda tab è chiamata “Campi”, qui intuitivamente verranno esposti tutti i campi prenotabili e il relativo costo orario. Cliccando sulla scritta costo orario verrà visualizzato un AlertDialog sul quale verranno visualizzati il costo del noleggio del campo in base alla fascia oraria.

Infine tappando il nome di un campo in lista si verrà portati nella activity che gestisce le prenotazioni relative a quel campo.

4.3.7 PRENOTATIONS ACTIVITY

Qui verranno gestite le prenotazioni relative ad un campo. Per di più vengono utilizzate due delle librerie esterne introdotte in 4.2.3.

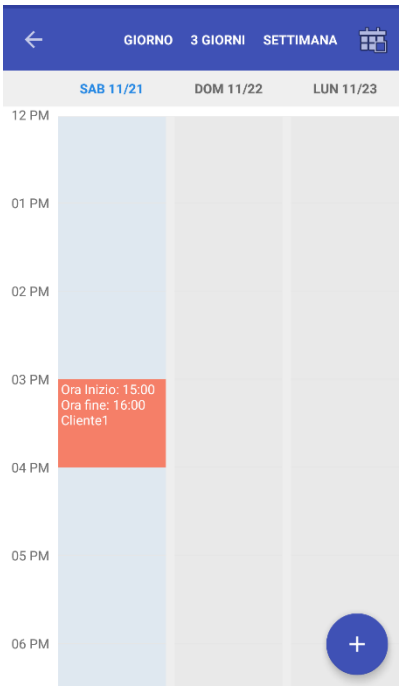


Fig. 20 – prenotation activity sreenshot

La schermata permette sia ad utenti anonimi sia a utenti loggati di visualizzare le prenotazioni già effettuate.

Un utente loggato invece può inserire una nuova prenotazione. Premendo quindi il Floating Action Button (il bottone in basso a sinistra) verrà chiesto di inserire nell'ordine:

- La data
- L'ora d'inizio
- La durata della prenotazione

Tutte le informazioni richieste verranno sottoposte ad un controllo di validità. Grazie alla libreria MaterialDateTimePicker è stato possibile imporre dei vincoli sulla scelta della data e dell'ora.

Le limitazioni sono state fatte in base agli orari del centro sportivo. Subito dopo che una data è stata scelta l'applicazione ritornerà un

messaggio di errore nel caso, in quella data, il centro risultasse chiuso. Anche il time picker dell'orario d'inizio è sottoposto a dei vincoli dinamici, siccome la sua libertà di scelta è stata limitata in base all'orario di apertura e di chiusura che il centro sportivo avrà nella data scelta poco prima. Infine la durata della prenotazione si assicurerà di rispettare gli orari di apertura e di chiusura.

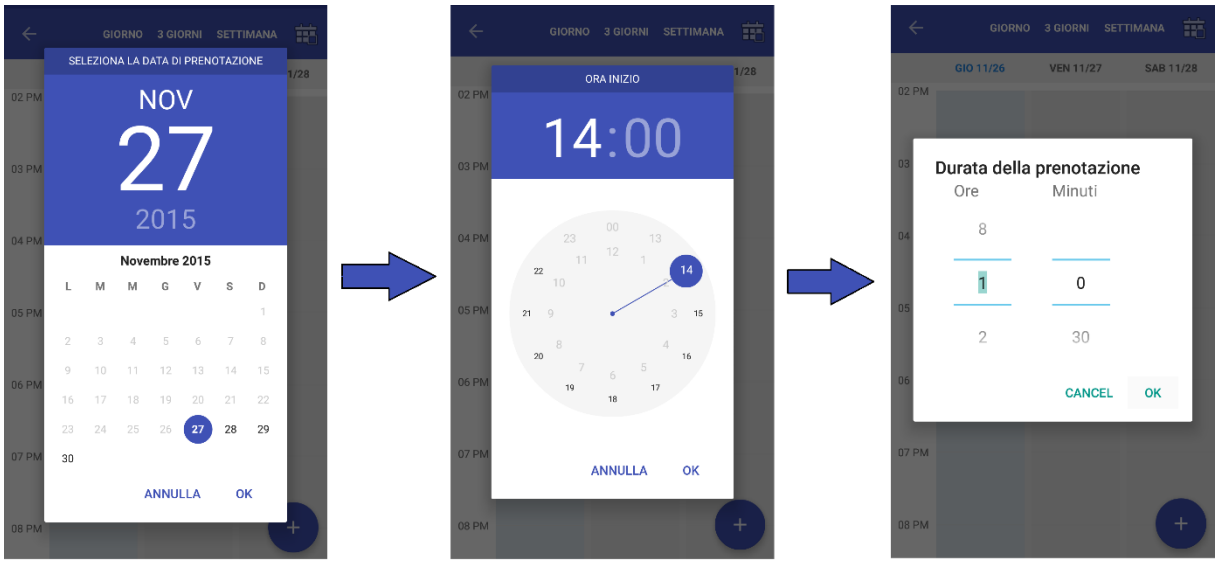


Fig. 21 – inserimento di una nuova prenotazione

4.3.8 MAPPA DELL'APPLICAZIONE

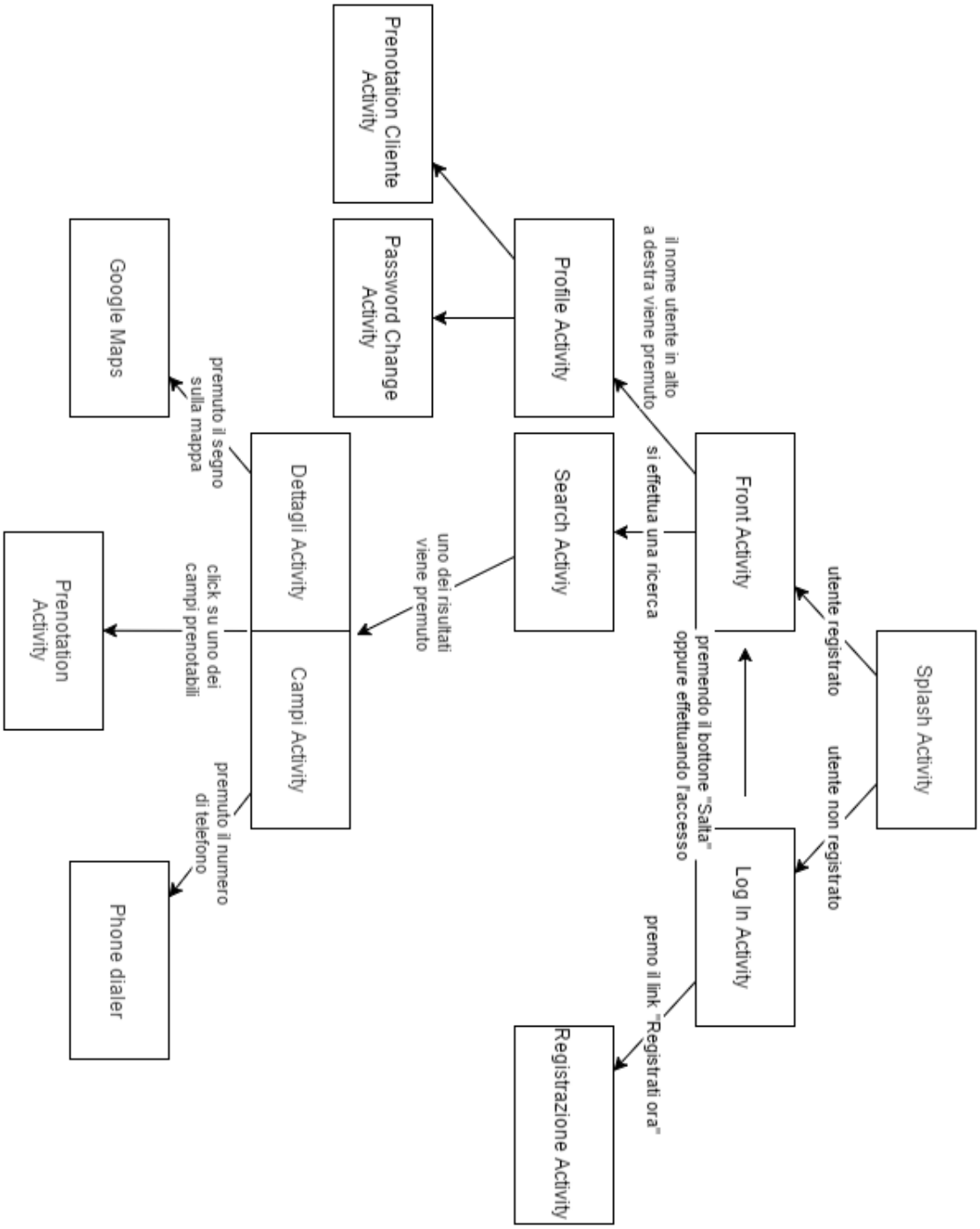


Fig. 22 – Mappa completa applicazione

5 CONCLUSIONI

In questa tesi si è cercato di sviluppare un sistema client-server che illustrasse come fosse possibile svilupparne uno partendo da zero.

Questo progetto è stato un buon spunto per approfondire le tematiche legate allo sviluppo in ambito servizi web e mobile, entrambi molto diffusi al giorno d'oggi. Data la mole di codice prodotto, è inoltre servito per approfondire i linguaggi utilizzati.

Tuttavia le performance attuali del servizio non sono tali da consentire un reale utilizzo dell'applicazione.

Ciò è dovuto soprattutto alla mancanza di un server dedicato al mantenimento del servizio. Durante tutto lo sviluppo dell'applicazione ci si è avvalsi di IIS7 come macchina virtuale dove effettuare dei test, ma anche di uno spazio sul web grazie al sito somee.com. Somee offre un pacchetto gratuito per hosting Asp.net, imponendo tuttavia molte limitazioni: sulla banda in entrata e in uscita, sulla grandezza del sito e del database rispettivamente 150 e 15 MB.

Altro aspetto sicuramente cruciale per una messa in produzione del servizio è la scalabilità che deve sicuramente essere migliorata.

Sul lato client le migliorie da apportare a mio avviso riguardano in larga parte la user experience che andrebbe modificata a seguito di uno studio mirato.

6 SVILUPPI FUTURI

L’applicazione, per poter diventare pienamente operativa e per poter essere pubblicata sul Play Store Android, avrebbe bisogno di alcune implementazioni aggiuntive.

Oltre alle modifiche di tipo perfettivo, il servizio dovrebbe beneficiare di espansioni sul lato social ormai un must-have nel mondo web in generale. Si pensi quindi ad una loro integrazione nel log-in.

La possibilità, per un utente registrato, di aggiungere un centro sportivo ad una lista di preferiti in modo da poterlo trovare più agevolmente in futuro potrebbe essere un’altra integrazione.

Altre aggiunte che sicuramente andranno fatte riguardano i centri sportivi. Si dovrebbe pensare ad un sistema per che permetta ad un centro sportivo di registrarsi, ma al contempo che ne garantisca la sua reale esistenza e la veridicità delle informazioni inserite. Per ridurre al minimo il rischio di account fake all’interno del db.

Sempre relativo ai centri sportivi, sarebbe interessante anche valutare l’implementazione di un sistema di notifiche push che avvisano i clienti nelle vicinanze (del cliente si conosce la città) quando vengono effettuate promozioni, eventi o sconti sul costo dei campi.

Altrettanto interessante sarebbe un sistema di geolocalizzazione che vada in automatico a cercare i centri sportivi più vicini all’utente, magari evidenziandoli su una mappa.

Questi riportati sono solo alcuni delle possibili aggiunte ad un servizio che risulta attualmente basilare; e quindi aperto ad una moltitudine di possibili miglioramenti.

Va in ultimo detto che, in effetti, il database sottostante l’applicazione era già stato pensato per alcune delle cose scritte qui sopra; come la gestione di dei preferiti, dell’integrazione con i social network. Sono già state create anche le api che permettono di inserire, modificare o

cancellare un centro sportivo dal db. Ma per problemi di tempistiche non è stato poi possibile sfruttare sul lato client queste implementazioni. Si è deciso tuttavia di non rimuoverle, ma di conservarle in vista degli sviluppi futuri.

Sperando che un giorno questa applicazione diventi finalmente operativa, ma soprattutto utile per qualcuno.