Sviluppo iOS

Il sistema iOS e la creazione di un'app

Alessio Rubicini

Elaborato per l'Esame di Stato 2020/21

Informatica - Sistemi e reti

Classe 5C - Articolazione Informatica

I.T.T. "G. e M. Montani" Fermo

31 maggio 2021

Indice

0.1 Premessa . . . 3
0.2 Ringraziamenti . . . 3
0.3 Introduzione . . . 3

1 Il sistema iOS **4**
1.1 Introduzione . . . 4
1.2 Architettura . . . 5
1.3 Sviluppo app . . . 6
1.3.1 Linee guida . . . 7
1.3.2 Strumenti e formazione . . . 7
1.4 Punti di forza . . . 8
1.4.1 Semplicità . . . 8
1.4.2 Integrazione . . . 8
1.4.3 Sicurezza e privacy . . . 9
1.4.4 Supporto . . . 10
1.5 Punti di debolezza . . . 11
1.5.1 Prezzo dei dispositivi . . . 11
1.5.2 Personalizzazione . . . 11
1.5.3 Altri svantaggi . . . 12
1.6 Differenze con Android . . . 12

2 Caso di studio: GameZen **13**
2.1 Introduzione . . . 13
2.2 Descrizione del sistema . . . 14
2.3 App iOS . . . 15
2.3.1 Funzionalità . . . 15
2.3.2 Architettura . . . 16
2.3.3 Tecnologie . . . 16
2.3.4 Compatibilità . . . 16
2.4 Servizi backend . . . 17
2.4.1 Architettura . . . 17
2.4.2 Tecnologie . . . 17
2.4.3 Funzionamento . . . 17
2.5 Database . . . 18
2.5.1 Progettazione concettuale . . . 18
2.5.2 Modello logico . . . 19
2.5.3 Implementazione . . . 19
2.6 UI/UX . . . 20
2.6.1 Layout . . . 20
2.6.2 Colori . . . 20
2.6.3 Feedback aptici . . . 20

2.6.4 Pop-up 20
2.7 Sicurezza e privacy 21
2.8 Sviluppo 22
2.9 Parti significative del codice 22
2.9.1 Lettura dei prodotti dal database 22
2.9.2 App iOS: gestione dello stato dell'app 23
2.9.3 App iOS: test unitari 24
2.10 Implementazioni future 24
2.10.1 Widget 24
2.10.2 Pagamenti in-app 25
2.10.3 Autenticazione biometrica 25
2.10.4 Sicurezza in rete 25
2.10.5 Accessibilità 25
2.10.6 Funzionalità avanzate per il negozio 25

3 Conclusioni **26**

Bibliografia **27**

0.1 Premessa

Il presente elaborato è stato realizzato dallo studente Alessio Rubicini della classe 5INC dell'Istituto Tecnico Tecnologico Montani di Fermo per l'Esame di Stato del secondo ciclo dell'anno scolastico 2020/21. L'elaborato è stato trasmesso dal docente di riferimento e consegnato nei tempi previsti, in ottemperanza a quanto previsto dall'ordinanza n.53 del 3 marzo 2021 del Ministero dell'Istruzione che disciplina le modalità di svolgimento degli Esami di Stato del secondo ciclo.

0.2 Ringraziamenti

Prima di procedere con la trattazione, voglio dedicare questo spazio del mio elaborato alle persone che hanno contribuito, direttamente o indirettamente, alla realizzazione dello stesso e che mi sono state vicine in questo percorso di crescita personale e professionale.
Vorrei innanzitutto ringraziare i miei docenti, in particolare la prof.ssa Carla De Benedictis che mi ha seguito nella realizzazione del presente elaborato, la prof.ssa Fabiola Farnese, il prof. Mauro Febi, il prof. Francesco Monaldi, il prof. Luigi Romagnoli e tutti gli altri docenti per il supporto fornitomi durante lo svolgimento del lavoro e durante l'anno scolastico. Un doveroso ringraziamento va ovviamente alla mia famiglia, senza la quale non avrei mai neppur cominciato il mio percorso, in particolare a mia sorella Ambra che ha contribuito direttamente a questo progetto. Infine una dedica speciale va a tutti i miei amici, alcuni dei quali anche miei colleghi, che mi hanno supportato durante il mio percorso di studi e che hanno sicuramente avuto un peso nel conseguimento di questo risultato; in particolare vorrei ringraziare tra i miei amici più cari Andrea, Daniele, Riccardo e Nicole che mi hanno aiutato in modi diversi e mi sono stati vicini sia nei momenti migliori che nei momenti più difficili. Grazie infinite a tutti voi.

0.3 Introduzione

In una società tecnologica e interconnessa come la nostra, dove i dispositivi mobili sono ormai alla portata di tutti e sempre più integrati nelle nostre vite, il ruolo degli sviluppatori mobile è ormai di fondamentale importanza. Sviluppare un'app mobile è per molti aspetti diverso dallo sviluppare un'app desktop o web: essa è installata sul dispositivo più personale dell'utente ed interagisce con il sofisticato hardware che lo compone. Elementi come il design dell'applicazione stessa e l'esperienza utente giocano un ruolo fondamentale e molto spesso sono determinanti nel giudizio dell'utente stesso.

L'obiettivo del presente elaborato è quello di fornire dapprima, nel contesto della materia sistemi e reti, un'introduzione teorica al sistema operativo iOS, illustrando la sua architettura e le sue caratteristiche, evidenziando i suoi punti di forza e di debolezza e le differenze con il sistema Android. Successivamente, nel contesto della materia informatica, viene presentata un'applicazione iOS realizzata nella pratica, in questo caso un'app per la gestione di un negozio di giochi da tavolo, esponendo le sue funzionalità, l'architettura dell'intero sistema, il design, l'esperienza utente e le tecnologie impiegate. Il codice sorgente è disponibile sulla repository GitHub del progetto che verrà resa pubblica dopo la consegna dell'elaborato al seguente link https://github.com/alessiorubicini/GameZen-iOS.

Quello dello sviluppatore mobile è un lavoro sempre più richiesto al giorno d'oggi ma soprattutto un lavoro sempre più importante, basti vedere l'impatto che hanno avuto alcune app in tutto il mondo contro la recente pandemia di COVID-19 [1]. Uno degli obiettivi di questo elaborato è proprio portare all'interesse di docenti e studenti questo mondo che al giorno d'oggi purtroppo non trova spazio nei programmi scolastici ma che, secondo la mia personale opinione, dovrebbe essere trattato almeno in termini basici, per poi essere proseguito in corsi universitari.

Capitolo 1

Il sistema iOS

1.1 Introduzione

Il 29 giugno 2007 Apple lanciava il primo iPhone che, con la sua interfaccia utente e il design elegante, ha trasformato non solo il business della telefonia mobile, ma anche l'economia di Internet e la società nel suo insieme. Dal punto di vista tecnologico però, il dispositivo in sé non era molto innovativo: lo schermo touch era già stato utilizzato nei precedenti prodotti come l'Apple Newton, l'interfaccia era basata sul già famosissimo iPod e in quanto a prestazioni i telefoni Nokia erano nettamente migliori. La vera rivoluzione di iPhone fu infatti il suo sistema operativo, inizialmente senza nome poi chiamato iPhoneOS e infine iOS.

L'idea iniziale di Steve Jobs per creare iOS era quella di ridurre il Mac, un'impresa di ingegneria di dimensioni epiche, o ingrandire l'iPod: dopo una accesa competizione interna tra il team del Macintosh, guidato da Scott Forstall e il team dell'iPod, guidato da Tony Fadell, l'idea del rimpicciolimento del già esistente Mac fu favorita da Jobs e questo portò numerosi vantaggi. Derivando infatti iOS dal Macintosh, avrebbero creato una piattaforma di sviluppo già familiare agli sviluppatori di terze parti, permettendo loro di creare applicazioni per iPhone con una discreta mole di lavoro [11].

Ad oggi iOS domina il mercato dei sistemi operativi mobile insieme ad Android. Ogni anno il sistema viene aggiornato, con aggiornamenti più grandi che introducono novità consistenti e con aggiornamenti più piccoli di bug fix, sicurezza e ottimizzazione, e nel corso del tempo è cresciuto in termini sia di funzionalità introdotte che di utenza.
Attualmente il 71% del mercato smartphone è coperto dal sistema Android mentre il 27% da iOS [5]. Tuttavia, al di là delle pure statistiche, l'iPhone è un dispositivo ampiamente conosciuto, venduto e utilizzato. Basta osservare l'analisi pubblicata da *Counterpoint Research* [8] la quale, a inizio aprile 2021, mostra una classifica governata da Apple, con gli iPhone 12 al primo posto tra i modelli di smartphone più venduti a gennaio 2021. Ma mentre qualcuno potrebbe amare iPhone semplicemente per la sua facilità d'utilizzo o l'interfaccia minimale, al di sotto della superficie si nasconde un sistema accuratamente progettato per fornire all'utente un'esperienza di alto livello.

iOS è la base di tutta la piattaforma mobile di Apple. Una delle sue caratteristiche peculiari, che come vedremo più avanti è uno dei suoi maggiori punti di forza, è il fatto che controlla tutti i differenti aspetti dell'hardware. Apple tramite iOS controlla l'intero ecosistema iPhone combinando software e hardware in un mix che garantisce le migliori prestazioni. Visto in un'ottica più ampia inoltre iOS si integra perfettamente con gli altri sistemi Apple, ma questi sono aspetti che verranno discussi successivamente.

1.2 Architettura

La piattaforma principale per iOS è l'architettura ARM, inizialmente solo a 32 bit e da iOS 11 necessariamente a 64 bit. Come macOS, iOS deriva da UNIX (che fa parte della famiglia di BSD) e usufruisce di XNU, un kernel ibrido e open-source basato su Mach e FreeBSD. Inizialmente iOS possedeva una versione del kernel superiore alla corrispondente di macOS. Nel tempo però i kernel di iOS e macOS si sono avvicinati e, soprattutto di recente, sembra che Apple stia gradualmente unendo i due sistemi operativi nel tempo.

In particolare, il sistema iOS è diviso in più livelli. Ogni livello si aggiunge ai servizi forniti dai livelli inferiori in modo tale che il livello più alto fornisca un set completo di servizi per gestire le comunicazioni ed eseguire le applicazioni. Le app iOS non comunicano direttamente con l'hardware sottostante, ma dialogano con esso attraverso una raccolta di interfacce di sistema ben definite.

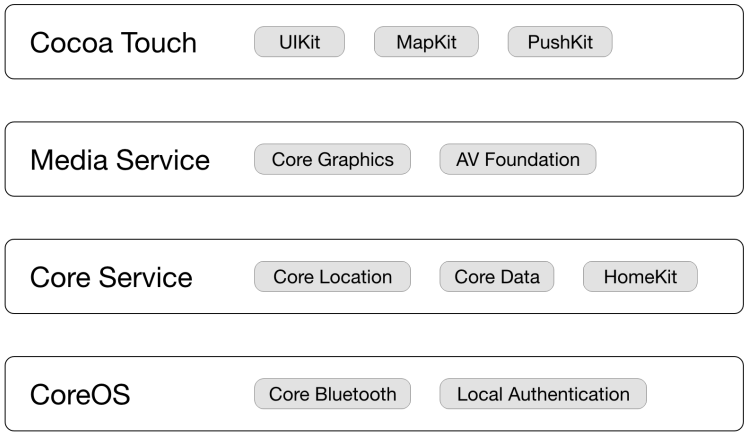


Figura 1.1: i 4 livelli principali dell'architettura iOS

Cocoa Touch

Questo è il livello più alto dell'architettura e si occupa della gestione e del riconoscimento dei movimenti sullo schermo, del touch e del multi-touch dell'utente, interpretando quindi in maniera corretta le gesture compiute da quest'ultimo. Oltre a gestire il touch dell'utente, questo strato si occupa di gestire funzionalità come l'accelerometro ed il giroscopio, riuscendo dunque a capire in che modo è orientato il dispositivo rispetto ad un asse orizzontale, e gestisce la gerarchia delle schermate, del multitasking, della fotocamera del dispositivo, della stampa, delle notifiche push e della condivisione dei file. A questo livello poi sono presenti i framework aggiuntivi utilizzati ad un più alto livello ovvero le API per lo sviluppo di app in ambiente Apple. Le librerie Cocoa Touch vengono infatti fornite con 2 framework:

- **Foundation** che fornisce un livello base di funzionalità per app e framework tra cui tipi dato base, archiviazione dei dati, gestione delle date, ordinamenti, filtri, ecc.;
- **UIKit** che fornisce gli oggetti utilizzati per la costruzione dell'interfaccia utente e per la definizione del comportamento delle app;

All'interno di questo livello figurano ovviamente svariate librerie tra cui MapKit per la gestione delle mappe e della coordinate geografiche, PushKit per la gestione delle notifiche e dei servizi VoIP, GameKit per il supporto al Game Center di Apple o EventKit per il controllo degli eventi nel calendario e altro.

Media Service

Questo livello gestisce tutto ciò che concerne la multimedialità: audio, video, immagini in vari formati, visualizzazione e modifica di file, gestione della libreria fotografica dell'utente. È utilizzato dagli sviluppatori per creare applicazioni e servizi che forniscono interfacce grafiche, animazioni e accesso agli elementi visivi nativi del dispositivo oltre che al comparto audio. La sua funzione principale è infatti l'implementazione delle librerie *Open-AL*, per la gestione dell'audio, e delle librerie *OpenGL-ES*, per la gestione della grafica 2D e 3D. Da iOS 4.0 qui si trova anche la gestione di AirPlay, la funzionalità che permette la riproduzione di contenuti multimediali su vari dispositivi Apple collegati in rete.

Core Services

Viene chiamato in questo modo perché fornisce servizi essenziali per le app, senza avere nulla a che vedere con la loro interfaccia grafica. Tra questi servizi troviamo la gestione delle connessioni di rete, la gestione dei database SQLite, la lista dei contatti, le preferenze di sistema dell'utente e altro. A questo livello si trova l'implementazione delle funzionalità di base più importanti come ad esempio la localizzazione, gli eventi, la telefonia e la gestione a basso livello dei media, oltre a una serie di librerie in C di base per la gestione di stringhe, date, URL, threads, ecc. Tutti questi servizi dipendono dal livello sottostante nonché ultimo livello dell'architettura, CoreOS.

CoreOS

È il livello più importante, il livello più basso dell'architettura di iOS, il nucleo del sistema operativo. Infatti si trova direttamente sopra l'hardware del dispositivo e gestisce tutte le sue funzioni vitali. Questo livello offre una varietà di servizi tra cui il networking di basso livello, l'accesso ad accessori esterni e i servizi fondamentali del sistema operativo quali il threading (POSIX threads), il networking (BSD sockets), lo standard I/O, i servizi DNS, i calcoli matematici, l'allocazione della memoria e la gestione del file system. Inoltre si occupa di gestire la batteria e la potenza erogata; per esempio quando la connessione wireless sull'iPhone viene disabilitata, questo livello va a spegnere la scheda Wi-Fi del dispositivo.

1.3 Sviluppo app

Inizialmente non erano previste app di terze parti su iPhone in quanto Steve Jobs voleva disporre di un sistema operativo privo di app esterne che potessero comprometterne il funzionamento, le prestazioni e la sicurezza. L'unica possibilità per gli sviluppatori era realizzare delle applicazioni web sfruttando il motore WebKit del browser Safari. Sotto però la pressione dei vertici dell'azienda e degli sviluppatori che ne compresero presto le grandi opportunità, verso la fine del 2007 Jobs cambiò idea. Questa decisione si concretizzò nella creazione di un SDK (Software Development Kit) nativo per iPhone, per permettere agli sviluppatori di creare app native, e nel lancio del famoso App Store nel luglio 2008. Ad oggi l'App Store di Apple contiene quasi 1,96 milioni di app disponibili al download contro le 2,87 del Play Store di Google. La qualità di iOS non è data solo dalla sua architettura e dal modo in cui funziona e si integra, ma si estende alle singole applicazioni e a come Apple si pone nei confronti degli sviluppatori che le realizzano.

1.3.1 Linee guida

Quando si entra nel mondo dello sviluppo iOS, uno dei primi strumenti che si è invitati a utilizzare sono le *Human Interface Guidelines* [4]. Come dice il nome, queste sono delle linee guida che Apple mette a disposizione degli sviluppatori per indicare loro come realizzare le proprie app in termini soprattutto di interfaccia utente (UI) ed esperienza utente (UX). Coprono diversi argomenti quali l'interfaccia grafica, l'architettura, l'interazione utente, l'uso delle funzionalità di sistema, il design, le icone, le immagini e le eventuali estensioni esterne dell'app come i widget. Le *Human Interface Guidelines* sono importanti non solo perché garantiscono all'utente un'app di buona qualità ma anche perché stabiliscono delle norme comuni grazie alle quali le applicazioni sono uniformi nel sistema Apple, utilizzando quindi componenti grafici familiari all'utente per facilitare l'utilizzo. Queste linee guida sono inoltre suddivise per sistema, definendo quindi apposite regole per iPhone, iPad, Mac, AppleTV e Apple Watch.

Apple però non definisce solo norme comuni per la realizzazione delle app ma anche vere e proprie regole che quest'ultime devono rispettare per poter essere pubblicate sull'App Store [2]. Ogni app infatti, prima di essere pubblicata, passa obbligatoriamente per un processo di revisione, solitamente dalla durata di 7 giorni durante i quali l'azienda si assicura che l'app rispetti le normative e sia efficiente ma anche e soprattutto sicura. Uno dei punti di forza dell'App Store e di iOS in generale è proprio lo stringente processo di revisione che ogni app deve passare per essere installata dagli utenti su iPhone, dal momento che l'App Store è l'unico modo legale e riconosciuto per la distribuzione in ambiente Apple.

1.3.2 Strumenti e formazione

Apple, oltre a fornire gli strumenti necessari, mette a disposizione degli sviluppatori una accurata documentazione delle proprie librerie e linguaggi, consultabile in ogni momento. Inoltre, l'azienda è molto attiva, soprattutto negli ultimi anni, nel settore dell'istruzione fornendo materiali come libri e corsi sia a insegnanti, per insegnare a programmare, che a studenti, per imparare a programmare. È doveroso far notare che Apple ha investito e continua a investire molto nella formazione di nuovi sviluppatori: un perfetto esempio è l'apertura nel 2016 della prima *iOS Developer Academy* a Napoli all'interno del nuovo campus dell'Università degli Studi di Napoli Federico II [3]. Il corso, per il quale è necessario un test d'ingresso, è completamente gratuito e ha l'obiettivo di preparare gli studenti nella creazione e gestione di applicazioni per le piattaforme dell'ecosistema Apple. Nella facoltà viene trattata anche amministrazione aziendale ed è previsto anche un percorso dedicato alla progettazione di interfacce grafiche. Gli studenti hanno l'opportunità di partecipare allo sviluppo di un'app intera, dalla progettazione all'implementazione, sicurezza, risoluzione dei problemi, archiviazione dei dati e utilizzo del cloud. Nell'anno accademico 2018/2019 sono arrivati studenti da più di 30 paesi diversi e una parte di questi studenti è stata selezionata per partecipare alla conferenza annuale per gli sviluppatori di Apple in California. Nel 2020, nonostate le difficoltà causate dall'emergenza pandemica, l'Accademia ha laureato quasi mille studenti da tutto il mondo, con oltre 400 idee di app di cui circa 50 già pubblicate sull'App Store.

Tutto questo dimostra quanto Apple tenga ai propri sviluppatori e alle straordinarie cose che essi realizzano. Nonostante questo però, di recente sono nate delle controversie di tipo economico: a settembre 2020 la nota azienda sviluppatrice di videogiochi Epic Games ha presentato una denuncia all'Antitrust nei confronti di Apple accusandola di monopolio. A detta di Epic infatti, Apple, applicando commissioni al 30% su quello che viene speso per acquisti in-app e abbonamenti nelle app iOS e limitando la distribuzione di applicativi al solo App Store, sfrutta la sua posizione dominante danneggiando sviluppatori e consumatori. L'azienda di Cupertino però ha risposto e ritiene che il controllo sul proprio negozio garantisca più equità, affidabilità e sicurezza. Al momento la controversia è ancora aperta ed entrambe le aziende stanno facendo valere la propria causa in tribunale.

1.4 Punti di forza

1.4.1 Semplicità

Il primo punto di forza di iOS è sicuramente la semplicità: dalla configurazione iniziale del dispositivo, all'utilizzo delle applicazioni, alle impostazioni, all'interfaccia utente fino alle funzionalità più tecniche, iOS è costruito in modo che gli utenti capiscano facilmente cosa possono fare e come farlo. Questa semplicità è frutto di anni e anni di lavoro e perfezionamenti ma soprattutto di una visione e di una filosofia di fondo che hanno contraddistinto Apple nel tempo.

Steve Jobs amava la semplicità. Un design chiaro, pulito e amichevole sarebbe diventato il segno distintivo dei prodotti Apple. In un'epoca non nota per i grandi designer industriali, le partnership di Jobs con Hartmut Esslinger negli anni '80 e successivamente con Jony Ive a partire dal 1997 hanno creato un'estetica ingegneristica e di design che ha distinto Apple dalle altre società tecnologiche e alla fine ha contribuito a renderla l'azienda più preziosa nel mondo. L'amore di Jobs per la semplicità nel design è stato affinato quando diventò praticante del buddismo e quando, tornato dall'India, andò a lavorare all'Atari nella progettazione di videogiochi con il suo amico Steve Wozniak. Jobs ha ripetutamente sottolineato in passato che il mantra di Apple è la semplicità e riteneva che una componente fondamentale del design fosse rendere i prodotti intuitivamente facili da usare. Quando nel 1984 si preparava a vendere il primo Macintosh, era letteralmente ossessionato dal problema del design e dei colori della confezione. Sia lui che Jony Ive, l'allora responsabile del design, erano convinti che il processo di spacchettamento dovesse essere una sorta di rito atto ad anticipare l'alta qualità del prodotto all'interno. Per Jobs la percezione del prodotto da parte del cliente era di estrema importanza: nella creazione dell'iMac insistette per aggiungere nella parte superiore del computer una maniglia, la quale era al servizio dell'utente e stava come a simboleggiare il permesso di toccare l'iMac.

1.4.2 Integrazione

Come accennato nell'introduzione, l'integrazione dell'intero sistema è uno dei suoi maggiori punti di forza. Steve Jobs una volta disse che il problema di Android contro iOS non era il fatto di essere aperto o meno, secondo lui il vero problema era la frammentazione rispetto all'integrazione. Affermava infatti che Android è troppo frammentato, ci sono troppe versioni diverse e troppi dispositivi diversi che mandano in confusione i consumatori. I dispositivi iOS invece non sono frammentati ma **verticalmente integrati**, Apple integra strettamente hardware e software e "funziona e basta". Ma che significa verticalmente integrati?

Jobs voleva trasformare dispositivi complessi come computer e smartphone in prodotti di mercato di massa e, per farlo, credeva che Apple dovesse strappare il controllo dei dispositivi all'utente. L'iPod è un buon esempio: le complessità della gestione di un lettore MP3 sono nascoste al consumatore dal fatto che il software iTunes gestisce tutta l'esperienza utente. I consumatori non possono acquistare brani da qualsiasi negozio online, ma l'iPod non si blocca quando della musica viene trasferita su di esso. Questo è l'aspetto pratico, la stretta integrazione di hardware e software rende il sistema più gestibile e prevedibile. Un sistema chiuso limita la scelta ma è più stabile e più affidabile, un sistema aperto lascia libero l'utente ma è molto più fragile e inaffidabile, questo è il prezzo della libertà.

L'integrazione non è intesa però solo limitatamente all'ambiente iPhone. iOS infatti si integra perfettamente con gli altri sistemi Apple macOS, iPadOS, WatchOS e TvOS andando a creare un unico

ecosistema. Se collegati a internet con lo stesso account iCloud, i dispositivi Apple vanno a formare una vera e propria rete personale di produttività, permettendo di effettuare innumerevoli operazioni che velocizzano il flusso di lavoro dell'utente come copiare un testo da iPhone a Mac, navigare su un sito su iPhone e continuare su Mac, sincronizzare i file sul cloud e tanto altro.

1.4.3 Sicurezza e privacy

Più volte Apple ha dimostrato e continua a dimostrare quanto tenga alla sicurezza e alla privacy dei propri utenti. In genere si sente dire che "iPhone e Mac sono più sicuri perché non prendono virus" ma questa è in realtà una leggenda metropolitana nata negli anni e non del tutto corretta. La verità infatti è che gli iPhone come i Mac e tutti gli altri dispositivi Apple non sono esenti da virus ma semplicemente sono più sicuri e meno diffusi rispetto per esempio a Windows e Android con conseguenti maggiore difficoltà e minore interesse nello sviluppo di virus appositi. La sicurezza è uno dei pilastri su cui si fondano i sistemi operativi Apple, sia lato hardware sia lato software.

Lato hardware, tutti i dispositivi Apple dispongono di funzionalità di sicurezza integrate direttamente nei componenti fisici. Tra questi il più importante è sicuramente il coprocessore chiamato *Secure Enclave*, utilizzato con il chip di sicurezza Apple T2 che interviene in ogni processo del sistema operativo (figura 1.2). Esso si occupa principalmente di tre funzioni: codifica dei dati, avvio protetto e rilevamenti biometrici. Per quanto riguarda la codifica, il chip include quello che viene chiamato *Dedicated AES Engine* ovvero un motore hardware dedicato alla crittografia AES-256 per potenziare la codifica durante la scrittura o lettura dei file e posizionato tra la memoria flash e la memoria di sistema principale, rendendo quindi la crittografia dei file altamente efficiente. Accanto a questo motore di crittografia troviamo anche il DMA (Direct Memory Access), un meccanismo che permette a sottosistemi come delle periferiche di accedere direttamente alla RAM senza passare per la CPU, generando quindi un solo interrupt. L'avvio protetto invece garantisce che all'avvio del sistema sia caricato solo il software autorizzato da Apple. Infine *Secure Enclave* permette a Touch ID e Face ID, rispettivamente impronta digitale e riconoscimento facciale, di garantire l'autenticazione sicura mantenendo i dati biometrici dell'utente privati e protetti. Gli utenti possono così contare sulla sicurezza garantita da codici e password più lunghi e complessi, accompagnata in molti casi dalla comodità di un'autenticazione immediata come quella biometrica.

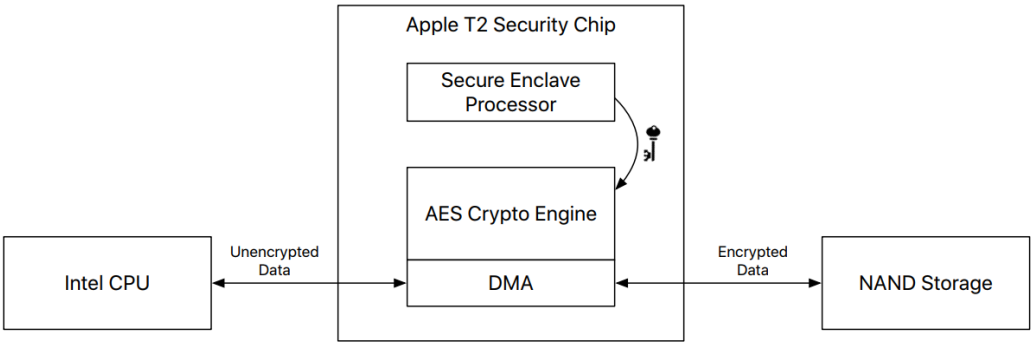


Figura 1.2: Panoramica del funzionamento del chip Apple T2 su Mac (2018)

Lato software, la sicurezza del sistema include i processi di avvio, gli aggiornamenti software e le operazioni in corso del sistema operativo. L'avvio protetto menzionato prima ha inizio nell'hardware e crea una sorta di catena lungo il software in cui ogni passaggio si assicura del corretto funzionamento di quello successivo prima di cedergli il controllo. Questo modello di sicurezza supporta anche

le varie modalità di recupero e aggiornamento dei dispositivi iOS, iPadOS e macOS. Il meccanismo di aggiornamento del software rende disponibili gli aggiornamenti per i dispositivi Apple in modo tempestivo, mettendo a disposizione software funzionante e sicuro. Il sistema di aggiornamento può anche impedire gli attacchi tramite downgrade[1], facendo in modo che i dispositivi non possano essere riportati a una versione precedente del sistema operativo, eventualmente vulnerabile agli attacchi di hacker, al fine di sottrarre i dati dell'utente. Infine, i dispositivi Apple sono dotati di protezione sia all'avvio sia durante l'esecuzione di processi, per garantirne l'integrità durante il funzionamento.

Tali protezioni variano sostanzialmente tra dispositivi iOS, iPadOS e macOS in base alle diverse funzionalità supportate e agli attacchi da cui devono quindi essere protetti. In ogni caso, i dispositivi Apple sono dotati di funzionalità aggiuntive che proteggono i dati dell'utente anche nel caso in cui altre parti dell'infrastruttura di sicurezza siano state compromesse. Tra queste troviamo per esempio metodi per la cancellazione immediata e totale dei dati a distanza in caso di furto o smarrimento del dispositivo. Questi strumenti di sicurezza ovviamente non sono applicati solo alle applicazioni ma anche ai servizi offerti da Apple come ID Apple (autenticazione), iCloud (archiviazione online) o Apple Pay (pagamenti). Tutto ciò è strettamente legato anche alla privacy degli utenti, un tema molto caro ad Apple. Una dimostrazione di quanto l'azienda tenga a questo problema di oggi si trova nell'ultimo grande aggiornamento di iOS lanciato a settembre 2020: è stata infatti introdotta una funzionalità la quale mostra una spia luminosa verde o arancione nella parte alta nel display quando un'app sta utilizzando rispettivamente la fotocamera o il microfono [7]. In questo modo gli utenti sanno quando un'applicazione sta utilizzando i dispositivi di input dell'iPhone e può agire di conseguenza. Ancor più importante è il famoso caso Apple-FBI [10]: i federali, in seguito ad un attentato nel dicembre 2015, chiesero ad Apple di fornire loro un software fatto appositamente per disattivare i meccanismi di protezione e sicurezza dell'iPhone e ottenere così accesso ai dati degli attentatori. Per l'azienda questo rappresentava però un pericoloso precedente legale: Apple rifiutò categoricamente definendo la creazione di una backdoor apposita per sbloccare qualsiasi iPhone una grave minaccia della privacy dei propri utenti, sarebbe come creare una chiave master capace di aprire milioni di lucchetti e rappresenterebbe un enorme pericolo se finisse nelle mani sbagliate.

1.4.4 Supporto

Un ultimo punto a favore di iOS è il supporto che riceve. Durante l'anno tutti i sistemi operativi di Apple seguono un preciso ciclo di aggiornamenti: a giugno alla WWDC (WorldWide Developer Conference) Apple presenta le nuove versioni dei sistemi operativi fornendo le beta necessarie agli sviluppatori per aggiornare le proprie app. La fase di beta testing dura circa 3 mesi e le versioni finali vengono rilasciate a settembre/ottobre in concomitanza con la conferenza di presentazione dei nuovi prodotti. Dopo il rilascio della versione principale durante l'anno seguono update più piccoli che aggiungono funzionalità, aumentano la sicurezza e risolvono bug riscontrati dagli utenti. Un esempio di ciò è la versione 13.5 di iOS, una versione intermedia rilasciata a maggio 2020 la quale conteneva le nuove API per il tracciamento dei contatti contro il COVID-19 [6]. Tutto ciò va ovviamente a favore degli utenti, i quali hanno sempre a disposizione l'ultima versione del sistema che introduce nuove funzionalità e migliora la sicurezza e la stabilità, andando a risolvere eventuali falle scoperte nell'utilizzo.

Stando ad un articolo di *VentureBeat* [9], dopo il passaggio di gestione da Steve Jobs a Tim Cook nel 2011, la frequenza degli aggiornamenti di iOS è aumentata dal 51%. Da una parte dietro questo aumento potrebbe esserci semplicemente la complessità del software che va aumentando negli anni, dall'altra parte potrebbe esserci il crescente numero di problemi che iOS ha riscontrato negli ultimi anni, soprattutto problemi riguardanti la codifica di particolari caratteri e la loro gestione in memoria che hanno causato importanti crash del sistema.

[1] Downgrade, letteralmente retrocessione: installazione di una versione precedente di un software.

1.5 Punti di debolezza

1.5.1 Prezzo dei dispositivi

È risaputo che i prodotti Apple in generale sono molto costosi e questo è uno dei motivi principali che spinge gli acquirenti a preferire altri marchi. Solitamente un nuovo iPhone ha un costo di minimo 500 € per il modello base fino a oltre 1300 € per un modello top di gamma. Negli ultimi anni infatti Apple ha iniziato ad inserire tra le nuove serie di iPhone un modello più economico in modo da raggiungere gli utenti meno propensi a spendere per uno smartphone, ma nonostante questo l'iPhone rimane comunque un prodotto di fascia alta e ci sono una serie di motivi per cui il suo prezzo è così alto.

Primo tra tutti la famosa integrazione hardware-software già citata più volte precedentemente. Apple progetta e ingegnerizza non solo l'hardware di ogni iPhone, ma anche il software e così facendo crea e controlla l'intera esperienza utente. Storicamente, concorrenti come Samsung hanno costruito i propri smartphone e utilizzato il sistema operativo Android di Google per eseguirli. L'integrazione attenta di software e hardware richiede più risorse e quindi, naturalmente, aumenta il prezzo del telefono.

Se si considera inoltre tutto ciò che un iPhone contiene e il fatto che ci sono dozzine di metalli di cui esso è composto, provenienti da ogni parte del mondo, prodotti su larga scala a mano e ognuno a volte a un costo umano piuttosto elevato, e che ci sono componenti altamente complessi come giroscopi, accelerometri e processori di ultima generazione incredibilmente compatti e potenti, l'iPhone è, in un certo senso, piuttosto economico. Quando si entra in un Apple Store e si leggono i prezzi non sembra affatto così ma è relativo: è piuttosto incredibile fare un passo indietro e considerare di cosa sono capaci i dispositivi che diamo per scontati e considerare il vero costo di portarli nelle nostre tasche. Pagare circa 700 € per uno smartphone, reso possibile dai ricercatori Apple in California, dai minatori in Bolivia, dai produttori in Cina e dagli sviluppatori di app in tutto il mondo, per citarne alcuni, improvvisamente sembra molto più ragionevole.

Ruolo importante nel prezzo è giocato anche dall'intero ecosistema Apple. L'azienda ha volutamente mantenuto il proprio ecosistema molto chiuso. Non è possibile, facilmente o legalmente, installare macOS su un computer diverso da un Mac come non è possibile ottenere iOS su altri dispositivi oltre a iPhone, iPod o iPad. Ciò significa che una volta entrati nel sistema, il prezzo per uscirne può essere piuttosto alto. Inoltre, le persone odiano il cambiamento, perciò una volta abituate a un sistema operativo non vogliono passare a uno nuovo, basta guardare come le persone reagiscono quando Microsoft apporta modifiche all'interfaccia di Windows 10. Gli utenti rimangono quindi bloccati nell'ecosistema e le loro uniche opzioni sono acquistare prodotti che non hanno concorrenti poiché solo i prodotti Apple hanno macOS o iOS. Ciò consente ad Apple di addebitare di più e avere ancora persone disposte a pagarlo.

Infine è ovvio che di un iPhone si paga anche l'immagine. Apple ha svolto un ottimo lavoro nel marketing dei propri prodotti nel corso degli anni ed è riuscita a posizionarsi all'apparenza come il marchio "cool" che tutte le persone "cool" acquistano. Far pagare molto i prodotti aiuta a mantenere questa immagine e crea un'esclusività attorno al marchio perché non tutti possono permettersi i prezzi che fai pagare.

1.5.2 Personalizzazione

Uno dei maggiori punti di debolezza sottolineati da tutti gli utenti è la poca personalizzazione. iOS infatti da sempre garantisce poca libertà di personalizzazione dell'interfaccia all'utente, limitandosi al cambio di immagine di sfondo, dei colori del display e piccole altre funzionalità legate più che altro all'accessibilità. Di recente però, con l'arrivo di iOS 14, le cose stanno cambiando e tramite varie nuove funzionalità come i widget e l'automazione degli script, gli utenti stanno creando veri e propri temi che

rendono iOS più originale, personale e diverso dallo stile di default che non tutti possono apprezzare. Esiste inoltre una procedura chiamata Jailbreaking la quale rimuove le restrizioni software imposte da Apple nei dispositivi iOS, permettendo così di installare software e pacchetti di terze parti, non firmati e autorizzati da Apple. In questo modo l'utente può installare applicativi che non sono presenti sull'App Store e modificare il sistema a proprio piacimento. Il Jailbreaking è legale ma solitamente è sconsigliato in quanto è un'operazione delicata e invalida la garanzia del dispositivo.

1.5.3 Altri svantaggi

- Sistema chiuso: un sistema chiuso pone delle limitazioni all'utente non permettendo o rendendo difficoltose operazioni normalmente consentite come per esempio il trasferimento di file multimediali;
- Scheda SD: al momento gli iPhone non dispongono di slot per scheda SD impedendo quindi di espandere la memoria oltre quella fornita di base dal dispositivo;
- Riparazioni costose: in molti casi riparare un iPhone, o un dispositivo Apple in generale, può essere molto costoso anche per danni di lieve entità;
- Dipendenza dall'interfaccia: l'interfaccia grafica di iOS è così semplice e facilmente utilizzabile che "crea dipendenza" fino a rendere difficoltoso tornare ad utilizzarne una diversa;
- Mancanze hardware: negli ultimi anni Apple ha cercato di semplificare gli iPhone rimuovendo periferiche come il jack da 3.5mm per le cuffie o l'impronta digitale in favore del riconoscimento facciale, ma questo non è apprezzato da tutti gli utenti come, per esempio, da chi non ha delle cuffie bluetooth e deve scegliere se caricare l'iPhone o ascoltare musica data la presenza di una sola porta;

1.6 Differenze con Android

Di seguito sono riportate le differenze tra il sistema iOS e il suo rivale Android.

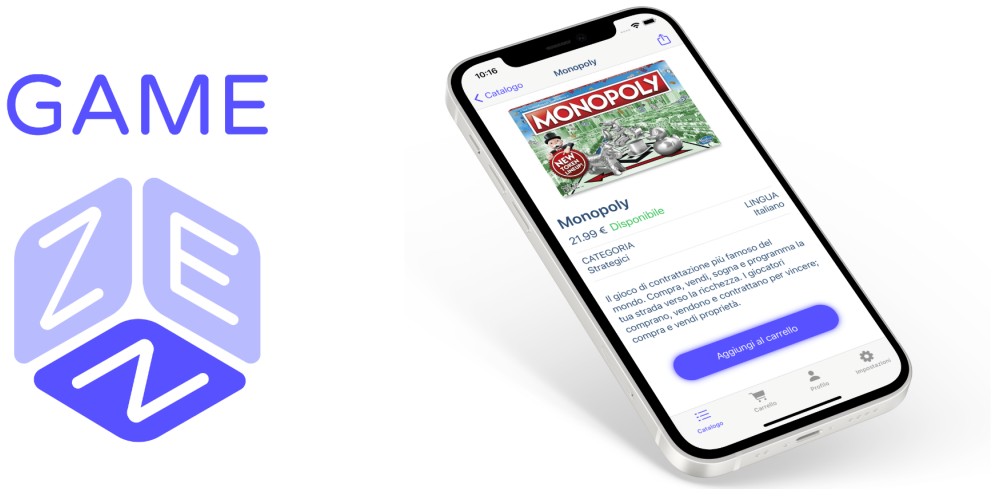
iOS	**Android**
Closed source con componenti open-source	Open-source
Disponibile solo su dispositivi Apple	Disponibile per molti produttori e dispositivi
App ottenute dall'App Store	App ottenute dal Play Store
Nessuno store alternativo all'App Store	Installabili store alternativi di terze parti
Strettamente integrato con ecosistema Apple	Meno integrazione con altri sistemi
Basato su un kernel ibrido	Basato su un kernel Linux-like
Le app sono scritte in Swift o Objective-C	Le app sono scritte in Kotlin o Java
Licenza proprietaria APSL / GNU GPL.	Licenza Apache 2.0 / GNU GPLv2.
Siri come assistente vocale	Google Assistant come assistente vocale
iCloud come cloud storage	Google Drive come cloud storage
Sviluppo app più facile e immediato ma costoso	Sviluppo app più complicato ma economico
Batteria con meno capacità ma ottimizzata	Batterie con grandi capacità ma poco ottimizzate
Generalmente più costosi	Generalmente più economici
Poco personalizzabile	Altamente personalizzabile
Continuamente aggiornato durante l'anno	Non sempre aggiornato causa frammentazione

È bene precisare che non esiste un sistema migliore di un altro: entrambi i sistemi hanno i propri punti di forza e debolezza e sono più adatti a uno specifico tipo di utenza piuttosto che ad un altro, sia in termini di utilizzo che di preferenze soggettive.

Capitolo 2

Caso di studio: GameZen

2.1 Introduzione



Come spiegato nell'introduzione, l'elaborato per la materia informatica consiste nella realizzazione di un'app iOS, quindi per dispositivi Apple, per la gestione di un negozio di giochi da tavolo. Riguardo il tipo di negozio (fisico/online) e il tipo di utenza a cui è rivolta l'applicazione (cliente/proprietario) è stata data piena libertà di scelta: di conseguenza si è ipotizzato che l'app rappresenti l'interfaccia mobile per il negozio online e che sia dedicata completamente all'esperienza del cliente, delegando le funzioni di amministrazione del negozio, da parte del proprietario e degli impiegati, ad altri eventuali software. Il nome scelto per l'applicazione è *GameZen*.

L'applicazione rispetta i seguenti requisiti funzionali riportati nella consegna:

- presentazione delle informazioni generali sul negozio e delle funzionalità offerte;
- registrazione di nuovi utenti, specificando i loro dati anagrafici;
- visualizzazione di tutti gli articoli presenti nel negozio;
- gestione di una pagina carrello;

In fase di progettazione sono state inoltre introdotte nuove funzionalità ritenute opportune nel contesto dell'applicazione o comunque utili ai suoi fini:

- visualizzazione dei dati dell'utente;
- suddivisione dei prodotti per categorie;
- ricerca dei prodotti nel catalogo per nome;
- inserimento di più indirizzi di consegna con scelta in fase di ordine;
- tracciamento degli ordini effettuati con possibilità di annullarli entro un certo limite di tempo;
- condivisione delle informazioni di un prodotto su altre app;

La lista completa dei requisti, funzionali e non, è stata riportata in un apposito documento di specifica dei requisiti.

2.2 Descrizione del sistema

L'intero sistema realizzato è formato da 3 componenti: app iOS, servizi backend[1] e database.

L'app iOS è l'applicativo principale che permetterà all'utente l'interfacciamento con il negozio per effettuare gli acquisti. È bene precisare che, nel contesto del presente elaborato, l'app non comprende funzionalità di pagamento in quanto, come specificato dalla consegna, il pagamento viene effettuato tramite bonifico in un secondo momento. Eventuali ipotesi di implementazione del pagamento in-app sono riportate nella sezione 2.10 mentre l'app iOS è descritta più dettagliatamente nella sezione 2.3. Tutte le immagini dell'applicazioni presentate in questo documento non sono mockup ma immagini reali del prodotto finale.

I servizi backend (API - Application Program Interface) non sono altro che l'interfaccia web tra l'app iOS e il database del negozio. Essi forniscono varie "funzioni" in rete al client per effettuare operazioni su di esso, sia di lettura, come per esempio la lettura dei vari prodotti disponibili o delle informazioni dell'utente, sia di scrittura come l'invio di ordini o l'aggiunta di indirizzi di consegna. Le API sono descritte più dettagliatamente nella sezione 2.4.

Il database è l'archivio sul quale i dati del negozio vengono memorizzati. La fase iniziale del progetto ha riguardato proprio il database con la progettazione concettuale e logica della base di dati. Il database è stato successivamente implementato tramite il DBMS MySQL e al suo interno sono stati inseriti dei dati, coerenti al dominio di progetto, per il successivo sviluppo dell'applicazione. Il database è descritto più dettagliatamente nella sezione 2.5.

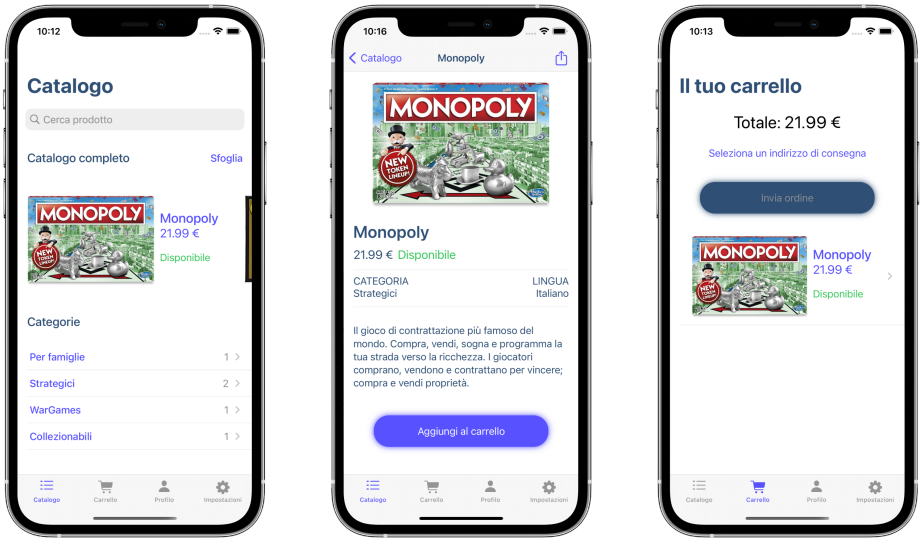
[1] Backend: l'insieme delle applicazioni e dei programmi con cui l'utente non interagisce direttamente ma che sono essenziali al funzionamento del sistema.

2.3 App iOS

In questa sezione viene discussa più dettagliatamente l'app iOS, illustrando il suo funzionamento, le tecnologie impiegate e l'architettura dell'applicazione stessa.

2.3.1 Funzionalità

L'app iOS fornisce tutte le funzionalità descritte precedentemente. Al primo avvio dell'applicazione viene mostrata la schermata di Onboarding ovvero di benvenuto per il nuovo utente, all'interno della quale viene brevemente illustrata l'applicazione e le sue funzionalità. Chiudendo la schermata di Onboarding, l'applicazione mostra quella di autenticazione: qui l'utente può effettuare l'accesso con le sue credenziali nel caso in cui sia già in possesso di un account o in caso contrario registrarsi. Nel caso in cui, in precedenza, l'utente abbia già effettuato l'accesso abilitando l'opzione `Rimani collegato`, l'accesso verrà effettuato automaticamente con le credenziali inserite senza visualizzare la schermata di autenticazione, velocizzando quindi l'accesso al negozio.



Una volta effettuato l'accesso, l'utente disporrà di una classica barra inferiore con 4 tab che permette la navigazione all'interno delle 4 schermate principali dell'applicazione:

- **Catalogo**: qui l'utente può consultare il catalogo osservando i prodotti disponibili, cercandone uno specifico tramite l'apposita barra di ricerca o navigando tra le categorie di prodotti. Premendo su di un prodotto può poi raggiungere la schermata del prodotto con tutte le informazioni necessarie e i bottoni per condividerlo o aggiungerlo al carrello;
- **Carrello**: qui l'utente può salvare e raggruppare degli articoli in attesa di effettuare l'ordine, per il quale è obbligatorio selezionare un indirizzo di consegna tra quelli aggiunti al profilo;
- **Profilo**: qui l'utente può consultare le informazioni del proprio account e agire su di esso gestendo gli indirizzi di consegna e gli ordini effettuati, oltre naturalmente a poter effettuare le classiche operazioni di uscita o eliminazione dell'account;
- **Impostazioni**: qui l'utente può gestire le proprie preferenze e trovare tutte le informazioni disponibili sul negozio;

2.3.2 Architettura

Internamente, l'app segue la classica architettura MVC (Model-View-Controller) declinata però in un modello più adatto al framework grafico SwiftUI, impiegato per lo sviluppo dell'app. Nello strutturare l'applicazione sono state perciò distinte 3 parti principali.

Modelli (Model): la struttura delle varie entità coinvolte all'interno dell'app come le informazioni dell'utente, i prodotti, gli ordini, ecc. Nello specifico, a livello implementativo sono state utilizzate le strutture del linguaggio Swift, conformi agli opportuni protocolli delle librerie Apple per la loro costruzione a partire dai dati in JSON[2] ricevuti dalle API.

Interfaccia (View): l'interfaccia grafica dell'applicazione che permette all'utente di interfacciarsi con essa. In determinate circostanze le schermate dell'app sono state presentate come modali nel caso di operazioni rapide come l'aggiunta di indirizzi di consegna.

Logica (Controller): la logica dell'applicazione ovvero gli oggetti responsabili della gestione dei dati nell'applicazione e della comunicazione con le API nonché del collegamento di modello e interfaccia.

2.3.3 Tecnologie

Dal punto di vista tecnologico, l'applicazione è stata realizzata utilizzando Swift come linguaggio di programmazione e SwiftUI come framework per la realizzazione dell'interfaccia e la gestione degli eventi.

Sono state inoltre impiegate varie librerie di terze parti:

- **Alamofire**: lo standard de-facto per l'invio di richieste HTTP su iOS;
- **SwiftUIX**: una libreria che fornisce componenti aggiuntivi per la creazione di interfacce grafiche con SwiftUI;
- **SwiftUIComponents**: una libreria contenente componenti sia frontend che backend per facilitare lo sviluppo con SwiftUI;
- **StatusAlert**: una libreria per la visualizzazione di pop-up stile Apple;

Informazioni più dettagliate sugli strumenti impiegati nello sviluppo sono fornite nella sezione 2.8.

2.3.4 Compatibilità

L'app in questione necessita, per essere installata, della versione 14.0 o superiore del sistema iOS la quale è compatibile con iPhone 7, iPhone 7 Plus, iPhone 8, iPhone 8 Plus, iPhone X, iPhone XS, iPhone XS Max, iPhone XR, iPhone 11, iPhone 11 Pro, iPhone 11 Pro Max, iPhone 12, iPhone 12 mini, iPhone 12 Pro, iPhone 12 Pro Max o iPhone SE. Non vengono inoltre utilizzate funzionalità particolari proprie di alcuni dispositivi e l'unico requisito strettamente necessario è la connessione a internet per l'interazione con il server del negozio.

[2] JSON (JavaScript Object Notation) è un formato adatto all'interscambio di dati fra applicazioni client/server. È basato sul linguaggio JavaScript Standard ECMA-262 3ª edizione, ma ne è indipendente.

2.4 Servizi backend

Come descritto precedentemente, i servizi backend ovvero le API (Application Program Interface) forniscono un'interfaccia all'applicazione iOS per interagire con il database del negozio. In questa sezione si illustrano la loro struttura, il loro funzionamento e le tecnologie impiegate.

2.4.1 Architettura

Le API si basano sul protocollo HTTP e sono organizzate in directory: ogni cartella corrisponde a un'entità della piattaforma, come ad esempio l'utente o l'ordine, e contiene, sotto forma di file `.php`, le varie funzioni disponibili per quella entità. Per gli utenti, per esempio, sono disponibili le funzioni di accesso e registrazione, per gli ordini le funzioni di lettura degli ordini inviati e di invio di un ordine, e così via.

In questo modo vengono fornite al client varie rotte, ognuna delle quali gli permette di effettuare una determinata operazione in relazione a un determinato elemento. L'app iOS invia quindi delle richieste HTTP alle API in base all'operazione che deve eseguire, le API a loro volta interagiscono con il database effettuando le dovute operazioni e restituiscono una risposta all'app la quale contiene il codice di stato HTTP, per verificare rapidamente il risultato, e, nel caso in cui debbano essere restituite delle informazioni, i dati prelevati in notazione JSON. Il client, all'arrivo della risposta, controlla il codice di stato per capire se la richiesta è andata a buon fine o meno, per poi effettuare tutte le opportune operazioni.

2.4.2 Tecnologie

Questa interfaccia è stata realizzata utilizzando il linguaggio PHP 7.4.12 ed è stata quindi implementata, come il database, su un server web. Nella sezione 2.8 sono riportate le tecnologie impiegate per lo sviluppo e il test del sistema.

2.4.3 Funzionamento

Come esempio per il funzionamento delle API, di seguito è illustrato il flusso di azioni che intercorrono nel caso in cui l'utente aggiunga un prodotto al proprio carrelo.

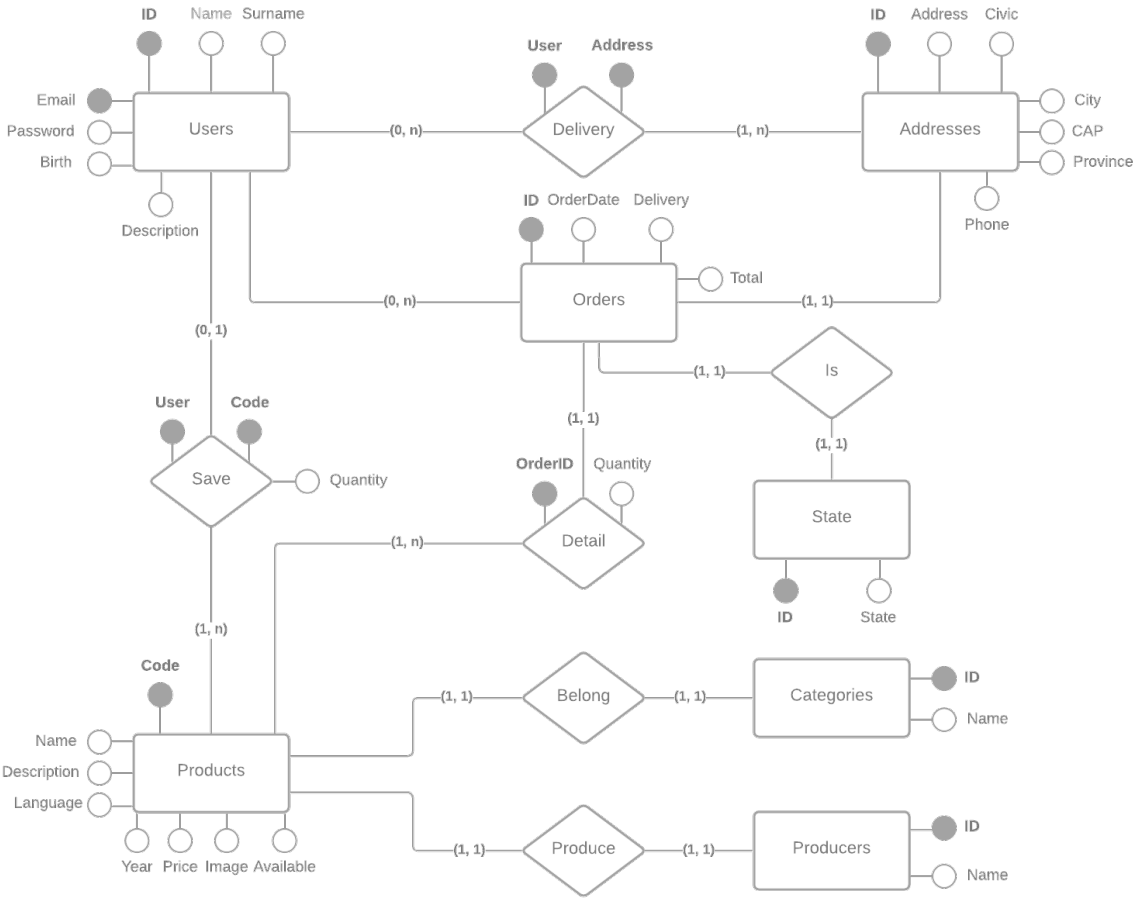
1. L'utente, dalla schermata del prodotto, clicca l'apposito tasto di aggiunta;
2. L'applicazione invia una richiesta HTTP all'apposito URL delle API contenente l'ID dell'utente e l'ID del prodotto da aggiungere;
3. Le API, dopo aver controllato che i dati siano stati passati correttamente si connettono al database e attuano le opportune procedure di sicurezza per rendere sicuri i dati in input ed evitare problemi di sicurezza;
4. Le API, dopo gli opportuni controlli descritti precedentemente, eseguono la query (comando) di inserimento nel database utilizzando i dati passati;
5. Se l'operazione è stata effettuata con successo, le API chiudono la connessione al database e ritornano una risposta positiva con codice 200 al client, in caso contrario ritornano una risposa negativa con codice 500;
6. L'applicazione riceve la risposta e la visualizza con un pop-up;

2.5 Database

Il database è una parte fondamentale del sistema in quanto su di esso vengono memorizzati tutti i dati del negozio. In questa sezione viene illustrata la sua struttura e le tecniche di progettazione impiegate oltre alle tecnologie utilizzate per l'implementazione.

2.5.1 Progettazione concettuale

La prima fase del progetto ha riguardato proprio la progettazione concettuale della base di dati. Si è quindi iniziato con un'analisi generale del problema e l'individuazione delle entità coinvolte che si sono concretizzate nella realizzazione di un modello entità-relazione.



È bene precisare che il modello riportato sopra non è identico per tutti i membri del progetto. Sono state infatti apportate delle piccole modifiche in base alle necessità di ognuno di noi considerate le diverse piattaforme di sviluppo.

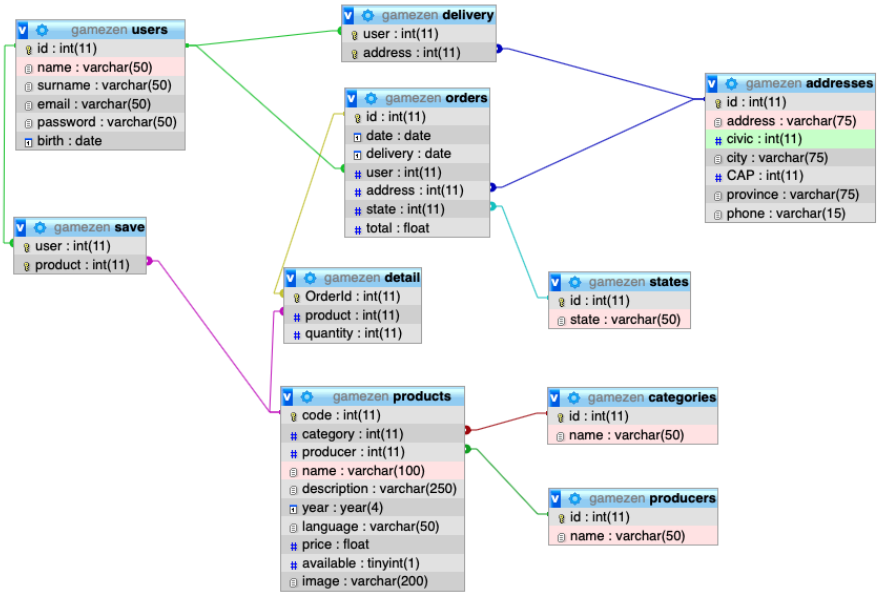
2.5.2 Modello logico

A livello logico, è stato scelto il modello relazionale per la realizzazione del database, il quale è composto dalle seguenti tabelle. Si noti che è stata impiegata la lingua inglese nella nomenclatura delle varie tabelle e attributi per mantenere una continuità con il codice, anch'esso scritto in inglese. La sottolineatura indica le chiavi primarie, utilizzate per identificare univocamente le tuple delle tabelle, l'asterisco * indica le chiavi esterne utilizzate per le relazioni tra tabelle.

Users (id, name, surname, email, password, birth): contiene tutti gli utenti registrati alla piattaforma, per ogni utente vengono riportati i dati anagrafici necessari al funzionamento del negozio;
Addresses (id, address, civic, city, CAP, province, phone): contiene tutti gli indirizzi di consegna che gli utenti hanno aggiunto;
Delivery (user*, address*): collega l'utente ai suoi indirizzi di consegna;
Products (code, name, description, year, language, price, available, image, category*, producer*): contiene tutti i prodotti venduti nel negozio, riportando le informazioni utili al cliente per l'acquisto;
Save (user*, product*): svolge la funzione di carrello collegando utenti e prodotti;
Orders (id, date, delivery, total, user*, address*, state*): contiene tutti gli ordini effettuati dagli utenti, è collegata ai vari prodotti inclusi nell'ordine tramite l'apposita tabella;
Detail (orderID*, product*, quantity): collega un ordine ai prodotti che contiene;
States (id, state): contiene i possibili stati in cui si possono trovare gli ordini effettuati;
Categories (id, name): contiene le varie categorie di prodotti disponibili;
Producers (id, name): contiene i vari produttori di giochi;

2.5.3 Implementazione

Una volta terminata la progettazione e vagliate tutte le ipotesi, il database è stato implementato a livello fisico. Anche stavolta, ogni membro del team di sviluppo ha effettuato l'implementazione sull'ambiente a lui più comodo per il successivo sviluppo dell'applicativo. Nel mio caso, il database è stato implementato tramite DBMS MySQL versione 5.7.32 integrato in ambiente di sviluppo locale MAMP versione 6.3. Di seguito è illustrato lo schema fisico del database.



2.6 UI/UX

Grande attenzione nel corso del progetto è stata posta all'interfaccia utente (UI) dell'applicazione iOS. In generale, è stato seguito il design del sistema iOS in modo da utilizzare componenti familiari all'utente, usufruendo di risorse messe a disposizione direttamente da Apple, come il set di icone *SF-Symbols* disegnato per integrarsi con il font San Francisco utilizzato di default sui sistemi Apple. Non mancano però componenti personalizzati, in modo da dare all'app un look originale e contraddistinguerla da una classica di sistema.

Strettamente legata all'interfaccia utente c'è anche l'esperienza utente (UX). Di seguito sono riportati alcuni punti fondamentali nella progettazione e nella creazione dell'esperienza utente di *GameZen*.

2.6.1 Layout

Gli elementi sullo schermo più utilizzati dall'utente, come il tasto di aggiunta al carrello o le tab per la navigazione tra le schermate, sono stati posti nella parte inferiore dello schermo in modo da essere facilmente raggiungibili anche con una sola mano e su schermi di grandi dimensioni.

2.6.2 Colori

I colori dell'applicazione sono stati scelti non solo per dare un look più originale all'app stessa ma anche per aiutare l'utente nel suo utilizzo. Sono stati infatti utilizzati colori come il blu per azioni neutre come l'aggiunta di un prodotto al carrello e colori come il rosso per azioni più pericolose come l'annullamento di un ordine. La palette colori dell'app è tendente al blu il quale, secondo alcuni studi, fa sentire l'utente al sicuro, migliorando l'esperienza, e viene associato al concetto di fiducia e affidabilità oltre ad essere facilmente utilizzabile anche da chi ha problemi alla vista.

2.6.3 Feedback aptici

Nell'applicazione si fa un uso opportuno di quelli che vengono chiamati feedback aptici. Per feedback aptico si intende tutta quella serie di funzioni atte a rendere l'esperienza touch dell'utente il più possibile vicina alla pressione di tasti fisici. Normalmente questa tecnologia è sviluppata con piccoli motori che, allo sfioramento dello schermo, rimandano una breve vibrazione. In GameZen infatti ogni azione, come l'invio di un ordine o l'aggiunta di un prodotto al carrello, termina con la generazione di un determinato feedback aptico in modo che l'utente capisca immediatamente, al di là del messaggio di errore a schermo, se l'operazione è andata a buon fine o meno. Da un punto di vista strettamente tecnico, la generazione di questi feedback è effettuata utilizzando un'apposita interfaccia messa a disposizione dalla libreria *SwiftUIComponents* sviluppata da me stesso.

2.6.4 Pop-up

Ogni volta che l'utente esegue un'azione, l'applicazione restituisce all'utente un feedback sia fisico, tramite i feedback aptici descritti precedentemente, sia visivo, attraverso la visualizzazione di un pop-up. Il pop-up visualizzato varia a seconda del successo o meno dell'azione eseguita:

- nel caso di risultato positivo, viene visualizzato un avviso di stato simile agli avvisi di default di iOS, con un'icona e un testo, che scompare automaticamente dopo qualche secondo, adatto quindi per avvisare l'utente senza interrompere il suo flusso di utilizzo dell'app;
- nel caso di risultato negativo, viene visualizzato un avviso che passa in primo piano e richiede la pressione dell'utente per essere chiuso, in modo da rendere chiaro cosa è andato storto e dove necessario eseguire eventuali azioni di correzione;

2.7 Sicurezza e privacy

Nello sviluppo del prodotto, una parte del lavoro è stata dedicata alla sicurezza e alla privacy dello stesso, con l'obiettivo di garantire all'utente un'esperienza sicura sotto tutti i punti di vista.

All'interno della base di dati, riguardo l'utente sono stati memorizzati solo i dati strettamente necessari al funzionamento del negozio, in relazione alle funzionalità attualmente implementate. In particolare, la password dell'utente non è stata memorizzata in chiaro ma è stato memorizzato il suo hash MD5[3], ottenuto in fase di registrazione nell'app tramite la libreria *CryptoKit* di Apple. In questo modo, sul database è presente solo una serie di caratteri alfanumerici piuttosto che la password in chiaro e nessuno a eccezione dell'utente può conoscere la reale password. Inoltre, considerando che quando l'app iOS scambia dati con i servizi di backend la comunicazione avviene su reti potenzialmente non sicure, ciò rappresenta una protezione da eventuali attacchi di tipo man-in-the-middle[4], garantendo la riservatezza e l'integrità della password nei dati in transito. Per quanto concerne le API, sono state attuate misure in particolare per prevenire vulnerabilità di tipo code injection[5] come la SQL injection o il cross-site scripting (XSS), evitando quindi che eventuali utenti malintenzionati accedano al database manipolando i dati inviati da loro in input.

Infine, nell'ambito della privacy, l'applicazione chiede sempre gli opportuni permessi all'utente prima di utilizzare le risorse del dispositivo come la connessione ad internet. Sono state inoltre inserite, sia nella schermata di registrazione sia nella schermata di impostazioni e informazioni, gli appositi link per arrivare alle eventuali privacy policy e condizioni d'uso dell'applicazione, le quali devono essere accettate dall'utente per completare la registrazione.

Future implementazioni per aumentare la sicurezza del sistema sono riportate nella sezione 2.10.

[3] L'hash è una funzione non invertibile che mappa una stringa arbitraria in una stringa di lunghezza predefinita.

[4] Attacco informatico in cui qualcuno segretamente ritrasmette o altera la comunicazione tra due parti che credono di comunicare direttamente tra di loro.

[5] Sfruttamento di un bug causato dall'elaborazione di dati non validi, viene utilizzata da un utente malintenzionato per introdurre codice in un programma vulnerabile e modificare il corso dell'esecuzione.

2.8 Sviluppo

L'applicativo è stato sviluppato su sistema macOS 11.3 e sono stati impiegati i seguenti strumenti:

- **Xcode**: IDE di Apple per lo sviluppo iOS;
- **Fastlane**: tool a riga di comando utilizzato per l'esecuzione rapida di tutti i test unitari;
- **Visual Studio Code**: editor per la scrittura di codice, utilizzato per la scrittura delle API;
- **MAMP**: una raccolta di software liberi per il funzionamento di server web sul sistema operativo macOS, utilizzato come ambiente di sviluppo web per l'hosting delle API e del database;
- **Postman**: un software per effettuare richieste di test alle API;
- **Git e GitHub**: rispettivamente strumento di controllo versione per il tracciamento delle modifiche apportate al codice e piattaforma di hosting di codice sorgente su una repository online;
- **Adobe XD**: software di progettazione utilizzato per la creazione dei mockup dell'applicazione;

2.9 Parti significative del codice

2.9.1 Lettura dei prodotti dal database

Codice PHP dei servizi backend: i prodotti vengono prelevati dal database e ritornati all'applicazione sottoforma di JSON con codice di stato HTTP 200.

```php
// Setup database interface
$db = new Database();

// Select all products from database
$result = $db->query("SELECT P.code, P.name, P.description, P.year, P.language, P.price,
    P.available, P.image, C.name AS 'category', producers.name AS 'producer' FROM
    products P, categories C, producers WHERE C.ID = P.category AND producers.id =
    P.producer");

// Transforming boolean value from tinyint to true|false
foreach($result as &$product) {
    if($product["available"]) {
        $product["available"] = true;
    } else {
        $product["available"] = false;
    }
}

// Close database connection
$db->close();

if(count($result) == 0) {
    // Return HTTP response
    header("HTTP/1.1 404");
} else {
    // Return data as JSON
    header("HTTP/1.1 200");
    echo json_encode($result, JSON_NUMERIC_CHECK);
}
```

2.9.2 App iOS: gestione dello stato dell'app

Codice Swift dell'applicazione iOS: l'intero stato dell'app è gestito da un oggetto `AppState` la cui classe è dichiarata nel codice sottostante. Questa dichiarazione include solo gli attributi e il metodo costruttore, gli altri metodi sono dichiarati in altri file tramite la keyword `extend` del linguaggio Swift che permette di estendere le funzionalità di una classe.

```
import Foundation
import SwiftUI
import Alamofire
import CryptoKit

// Identifies the app state and manages all the data
class AppState: ObservableObject {

    // User account properties
    @Published var user: User? = nil
    @Published var showRegistrationSheet = false
    @Published var isLogged: Bool = false
    @Published var alert = (false, "", "")

    // Delegating cart and catalog management responsibility to other objects
    @Published var cartManager: CartManager
    @Published var catalogManager: CatalogManager

    // Class constructor
    init() {

        // Initialize properties
        self.cartManager = CartManager()
        self.catalogManager = CatalogManager()

        // Check if "keepConnected" flag is true for automatic login
        if UserDefaults.standard.bool(forKey: "keepConnected") == true {

            // Get user's credentials from User Defaults local storage
            let email = UserDefaults.standard.string(forKey: "email") ?? "-"
            let password = UserDefaults.standard.string(forKey: "password") ?? "-"

            // Call login method with user's credentials
            self.login(email: email, password: password)
        }

    }


}
```

2.9.3 App iOS: test unitari

Codice Swift dell'applicazione iOS: di seguito è riportato uno dei test unitari scritti per testare l'applicazione, in particolare viene testata la lettura dei prodotti dai servizi di backend.

```
import XCTest
import Alamofire

class CatalogTests: XCTestCase {

    func testProducts() throws {

        // Test async expectation
        let e = expectation(description: "Get products")

        // Make call to API
        AF.request(API.getAllProducts.rawValue, method: .get)
            .response { response in

                do {

                    // Parse user info as JSON to Swift struct
                    let _ = try JSONDecoder().decode([Product].self, from: response.data!)
                    XCTAssert(true)
                    e.fulfill()

                } catch {
                    XCTAssert(false)
                    print(error)
                    e.fulfill()
                }
            }

        // Wait for asynchronous api call
        self.waitForExpectations(timeout: 5.0, handler: nil)
    }
```

2.10 Implementazioni future

In questa sezione sono riportate eventuali funzionalità implementabili atte a migliorare l'applicazione e il sistema in generale.

2.10.1 Widget

Un punto a favore dell'applicazione sarebbe rappresentato dal supporto ai widget[6], recentemente introdotti con la versione 14.0 di iOS. Si potrebbero implementare vari widget di diverse dimensioni tramite i quali l'utente può visualizzare le ultime offerte nel negozio o controllare rapidamente lo stato di un ordine, il tutto direttamente dalla schermata principale dell'iPhone senza entrare nell'applicazione.

[6] I widget sono una sorta di scorciatoia da inserire sulla home del dispositivo che permette visualizzare delle informazioni in maniera veloce e immediata, senza il bisogno di passare dalla rispettiva applicazione.

2.10.2 Pagamenti in-app

Vista la scomodità di effettuazione di un bonifico per l'accettazione e la spedizione dell'ordine, sarebbe eventualmente implementabile il servizio *ApplePay* per il pagamento degli ordini direttamente nell'app, usufruendo quindi dell'account iCloud dell'utente e dei metodi di pagamento ad esso collegati.

2.10.3 Autenticazione biometrica

Si potrebbe usufruire dei dispositivi di autenticazione biometrica installati sugli iPhone per permettere all'utente di accedere all'applicazione in maniera veloce e allo stesso tempo sicura. La stessa funzionalità si potrebbe inoltre sfruttare anche per operazioni delicate come l'invio o l'annullamento di un ordine.

2.10.4 Sicurezza in rete

Per aumentare il livello di sicurezza dei servizi backend, si potrebbe usufruire del protocollo HTTPS con TLS[7] per garantire una connessione privata, mediante algoritmi di crittografia a chiave simmetrica, autenticazione, tramite algoritmi di crittografia a chiave pubblica, e affidabilità, con controlli di integrità sui dati inviati atti ad avere la certezza che non vengano alterati durante la trasmissione. Il certificato TLS potrebbe essere ottenuto a pagamento da un'autorità certificante oppure potrebbe essere generato come proprio certificato (self-signed) utilizzando ad esempio librerie *openssl*: in questo secondo caso però si potrebbe assicurare la riservatezza delle connessioni mediante cifratura ma non l'autenticazione del proprietario.

2.10.5 Accessibilità

Tramite appositi strumenti messi a disposizione da Apple, si potrebbe migliorare l'accessibilità dell'applicazione: per esempio la possibilità di riprodurre a voce le informazioni di un prodotto o di un'ordine effettuato potrebbe aiutare gli utenti non vedenti.

2.10.6 Funzionalità avanzate per il negozio

La piattaforma in generale potrebbe poi essere migliorata con l'aggiunta di nuove funzionalità come le seguenti:

- possibilità di creare varie liste dei desideri in cui salvare i prodotti da aggiungere successivamente al carrello;
- gestione di un saldo buoni regalo stile Amazon;
- invio di notifiche per ogni nuovi aggiornamento degli ordini effettuati;
- creazione di una sezione offerte speciali;
- sistema di newsletter per segnalare le offerte speciali via mail;
- analisi delle preferenze degli utenti per fornire consigli sui prodotti da acquistare;
- integrazione di funzionalità di machine learning per il riconoscimento di un prodotto dalla fotocamera dell'iPhone (eventualmente sviluppabile tramite gli appositi tool messi a disposizione da Apple);

[7] Transport Layer Security (TLS) e il suo predecessore SSL sono dei protocolli crittografici che permettono una comunicazione sicura dalla sorgente al destinatario (end-to-end) su reti TCP/IP (come ad esempio Internet) fornendo autenticazione, integrità dei dati e confidenzialità.

Capitolo 3

Conclusioni

In conclusione, nel lavoro svolto è stato presentato il sistema iOS e ne è stata analizzata l'architettura, composta da diversi livelli integrati tra loro in una combinazione di hardware e software che garantisce le migliori prestazioni e la massima compatibilità tra le applicazioni degli sviluppatori e il sistema operativo stesso. Sono stati evidenziati i suoi punti di forza, come la sicurezza e la già citata integrazione, i suoi punti di debolezza, alcuni dei quali ormai noti a tutti come l'elevato prezzo di riparazione o la limitata personalizzazione, e le differenze con il suo rivale Android. Nella seconda parte dell'elaborato, è stata presentata l'applicazione iOS realizzata. In particolare, sono state coperte con un buono approfondimento tutte le componenti e le tecnologie che entrano in gioco nella realizzazione di una piattaforma come questa, dalla base di dati per la memorizzazione delle informazioni ai servizi backend per l'interfacciamento con essa passando per l'architettura dell'app iOS fino alla sua interfaccia ed esperienza utente. È opportuno sottolineare il fatto che l'intero sistema, in quanto elaborato per l'Esame di Stato, è stato realizzato interamente da un singolo studente (ad eccezione della progettazione della base di dati) e in un tempo di circa 30 giorni, meno considerando le ultime fasi di test, ottimizzazione, refactoring e stesura della presente relazione. In un contesto reale, lo sviluppo di un'app, soprattutto mobile, ha una durata di mesi e mesi di tempo e prevede un team di sviluppo formato da più soggetti specializzati in diversi campi.

Questo progetto non è solo il risultato del mese di lavoro a esso dedicato ma è in realtà frutto di mesi e mesi di studio ed esercitazione pratica sia in ambito scolastico che extrascolastico. Mi sono impegnato a fondo nello studio di questo campo dell'informatica, ho realizzato altre applicazioni prima di questa, ho partecipato a webinair di importanti aziende e ascoltato esperti in merito. Sono personalmente molto soddisfatto del risultato finale e sarò felice di ricevere un feedback, positivo o negativo che esso sia, da chi ha visionato questo lavoro. Lo sviluppo mobile, al momento, è un argomento che troppo spesso non trova spazio all'interno delle nostre scuole, spesso per mancanza di conoscenze da parte dei docenti o per mancanza di strumenti da parte della scuola stessa o semplicemente perché si rimane ancorati agli stessi argomenti del passato. Esso rappresenta però uno dei tanti lavori del futuro che stanno nascendo nel campo IT e sta crescendo di importanza, visto soprattutto lo straordinario sviluppo tecnologico, economico e sociale che stiamo vivendo del quale gli smartphone sono ormai protagonisti. Voglio quindi concludere questo elaborato invitando gli studenti come me a interessarsi a nuovi campi come questo e i docenti a prenderli in considerazione, a impararli, a osservarne e comprenderne le varie sfaccettature e a trasmetterli a loro volta, sia nella teoria che nella pratica, ai propri studenti con la professionalità e la passione di sempre.

Bibliografia

[1] AgendaDigitaleEU. *L'app covid inglese ha bloccato tanti contagi, quale lezione per Immuni*. 2021.

[2] Apple. *App Store Review Guidelines - Apple Developer*. 2021.

[3] Apple. *Apple opens first iOS Developer Academy in Naples*. 2016.

[4] Apple. *Human Interface Guidelines - Design - Apple Developer*. 2021.

[5] gs.statcounter.com. *Mobile Operating System Market Share Worldwide*. 2021.

[6] iPhoneItalia. *Con iOS 13.5 arriva la nuova API di notifica esposizione COVID-19*. 2020.

[7] Il Fatto Quotidiano. *Apple iOS 14: due puntini di notifica per difendere la nostra privacy*. 2020.

[8] Counterpoint Research. *Apple Takes 6 Spots in Top 10 Best-selling Models' List for Jan*. 2021.

[9] VentureBeat. *Apple's iOS update frequency has increased 51% under Cook's management*. 2018.

[10] Wikipedia. *FBI–Apple encryption dispute*. 2021.

[11] Wikipedia. *iOS*. 2021.