Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea in Informatica

Garzone
Il link tra PA, cittadini e commercio locale

Relatore: Prof. Denaro Giovanni

Tutor Aziendale: Fumagalli Aliosha

Relazione di tesi:
Giuseppe Facchi
Matricola 845662

Anno Accademico 2020-2021

A mamma, papà e Gabriele per avermi accompagnato tenendomi per mano durante tutto il mio percorso.

A Siria, perché senza di lei probabilmente non sarei mai riuscito a rialzarmi nei momenti di difficoltà

A Ruben e Pietro, fedeli e fondamentali compagni di viaggio per avermi fatto sempre sorridere e avermi aiutato a superare gli ostacoli più difficili.

A tutte le persone, compagni e professori, incontrate in Università che hanno contribuito a farmi amare questa Scienza più di quanto lo facessi già.

Al mio tutor aziendale Ali, che mi ha insegnato tutto quello di cui avevo bisogno per una solida crescita professionale e che mi ha concesso di esprimere al meglio in un piacevole contesto di lavoro.

Indice

Capitolo 1

Introduzione

Garzone è una soluzione informatica rivolta alla pubblica amministrazione, in particolare ad enti comunali, al servizio dei cittadini e del commercio locale. L'idea è nata dalla necessità di rilancio del commercio locale, oppresso dal progresso dei giganti tecnologici e dalla pandemia. Affiliata all'iniziativa "Soldiarietà Digitale" promossa dal Ministero dell'Innovazione Tecnologica[1], successivamente si è poi evoluta sulla base dei consigli di varie attività commerciali e di amministrazioni pubbliche. Il progetto, dopo la sua pubblicazione, ha inoltre ricevuto il "Premio Top of the Pid - Restart" [2], promosso dalle Camere di Commercio italiane, come miglior modello di business 4.0 per il rilancio del commercio locale. L'iniziativa premia i migliori progetti innovativi che possono agevolare il rilancio dell'economia per uscire dalla profonda crisi provocata della diffusione del Covid-19.

Figura 1.1: Premio Top of the Pid - Restart

1.1 Specifiche progettuali

La finalità principale di Garzone riguarda il soddisfacimento della necessità dei commercianti di poter aggiornarsi, tramite un primo approccio, al commercio online. La soluzione,

Figura 1.2: Logo di Garzone

per raggiungere l'obiettivo, deve permettere a un titolare di poter gestire con semplicità il proprio catalogo di prodotti venduti e servizi offerti e poterlo condividere online con la cittadinanza. Inoltre, in un contesto dove la presenza online è strettamente necessaria per far conoscere al pubblico la propria attività, un titolare deve avere a disposizione tutti gli strumenti per venire contattato e per far sapere dove svolge la sua attività tramite la costituzione di una vetrina virtuale. Fondamentale è poi l'interazione tra cittadino e commerciante, la quale deve essere garantita mediante l'implementazione di un'interfaccia per scambiarsi messaggi di chat e per una corretta gestione degli ordini di ritiro e di consegna a domicilio, eventualmente con pagamento online.
Di enorme rilevanza, per avere una buona accessibilità alla piattaforma da parte dell'utenza, è inoltre previsto l'utilizzo di linee guida grafiche ispirate agli standard pubblicati da AgID (Agenzia per l'Italia digitale) [3]. Gli enti comunali coinvolti dovranno invece avere a disposizione degli strumenti di controllo, per verificare il funzionamento dell'infrastruttura e il suo corretto utilizzo da parte dei commercianti.
Inoltre sarà per loro possibile comunicare con l'utenza mediante l'apposita sezione dell'applicativo a loro dedicato. Infine dovrà essere prevista, tramite l'implentazione dedicata di una piattaforma, l'amministrazione di tutta l'infrastruttura inizialmente da parte dell'azienda, ma in futuro da terzi.

1.2 Organizzazione del lavoro

Inizialmente lo stage prevedeva la presenza presso la sede aziendale dove il coordinamento avveniva mediante riunioni giornaliere prefissate tra il team di sviluppo ed il tutor aziendale, generalmente il mattino e a fine giornata. Nel caso emergessero problematiche, causate in genere dall'inesperienza, risultava molto semplice interfacciarsi tra colleghi presenti in loco. Con il riaggravarsi della pandemia il lavoro è stato riorganizzato in modalità full-remote tramite l'utilizzo della piattaforma Clickup. Le riunioni venivano svolte giornalmente il mattino e prevedevano in aggiunta un momento dove confrontarsi sui problemi emersi durante l'orario di lavoro. Dopo l'inserimento del collega Agazzi, il tutor aziendale ha previsto una suddivisione del lavoro tramite la definizione di task precisi, eseguibili solo dopo una formazione collettiva da lui fornita. A fine giornata era prevista una chiamata di allineamento tra il team di sviluppo per aggiornare l'un l'altro sugli sviluppi effettuati

durante la giornata e per chiarire problematiche non risolte singolarmente. Il testing delle implementazioni veniva eseguito dal tutor aziendale, il quale si occupava di segnalare eventuali problematiche e possibili correzioni implementabili.

Capitolo 2

Analisi dei requisiti

Un sistema informatico deve, in genere, risolvere un determinato problema relativo ad una categoria di utenza. Per raggiungere l'obiettivo deve quindi fornire un certo numero di funzionalità relative alla gestione delle informazioni, possedendo caratteristiche precise di efficienza e affidabilità. I requisiti riguardano per l'appunto la comprensione e la descrizione del problema da risolvere mediante l'implementazione di funzionalità e di qualità desiderate per il sistema.

2.1 Stakeholders

Si definisce *stakeholder* una qualsiasi entità dotata di comportamento che interagisce con il sistema informatico in questione. Quest'ultimo è inoltre considerato anch'esso uno stakeholder quando interagisce con altri sistemi informatici, nel caso del progetto Garzone quando interagisce, ad esempio, con il sistema di pagamento adottato.

2.1.1 Attori primari

Un'entità che utilizza direttamente la soluzione per perseguire i suoi obiettivi è considerata un attore primario. In Garzone gli attori primari sono rappresentati da:

Negozi Entità costituite da un punto vendita fisico in un determinato comune registrato sulla piattaforma. Sono rappresentati da un titolare o referente che ha il compito di gestire il proprio catalogo e interagire con i clienti finale utilizzando le varie funzionalità

Comuni Organi della pubblica amministrazione che delegano un funzionario facente parte della giunta eletta per la gestione e il controllo dei negozi registrati alla piattaforma. Il loro ruolo è meramente di supervisione, possono infatti abilitare o disabilitare la

visualizzazione di negozi nel marketplace. Inoltre hanno la possibilità di scambiare comunicati con i negozi.

Amministratore Persona o gruppo di persone organizzate per la gestione e la supervisione complessiva della piattaforma. L'amministrazione avviene mediante l'interazione con negozi, comuni e clienti che necessitano di supporto o di interventi particolari, come la disabilitazione per condotte scorrette. In Garzone questa mansione è per ora affidata all'azienda stessa, ma in futuro è prevista una delegazione a terzi.

2.1.2 Attori finali

Sono definiti attori finali coloro che usufruiscono del servizio messo a disposizione per raggiungere i propri obiettivi. In Garzone è previsto un solo attore finale: il cliente.

Cliente I clienti rappresentano il bacino di utenza della piattaforma e hanno la possibilità di interagire con i negozi per raggiungere i propri obiettivi, quali l'acquisto di beni/servizi, la comunicazione diretta con i corrispondenti referenti, la richiesta di preventivi, la prenotazione di appuntamenti, la visualizzazione dei cataloghi e altro.

2.1.3 Attori di supporto

Generalmente gli attori di supporto offrono alla soluzione servizi esterni necessari per il suo corretto funzionamento. Spesso sono sistemi informatici, ma possono anche essere rappresentati da società/organizzazioni o persone fisiche

Servizio di autorizzazione dei pagamenti In Garzone è prevista un'interfaccia ad un servizio di autorizzazione dei pagamenti per autorizzare i pagamenti correttamente ed evitare tentativi di frode. Inoltre ha il compito di registrare le transazioni dirette tra Clienti e Negozi.

Servizio di autenticazione degli utenti Per la corretta gestione degli utenti e la prevenzione di attacchi informatici sui dati di registrazione, Garzone si interfaccia a un servizio esterno di registrazione e autenticazione degli utenti. Esso ha il compito di comunicare con il sistema informatico e garantire il corretto flusso di registrazione, accesso e gestione delle utenze.

Servizio di geolocalizzazione Per offrire un'esperienza utente migliore, in Garzone è prevista l'implementazione di un'interfaccia ad un servizio di geolocalizzazione, il quale ha

il compito di interpretare indirizzi geografici forniti dal sistema e restituire delle coordinate geografiche precise.

2.2 Requisiti

Un requisito è una capacità o una condizione a cui il sistema, e più in generale, il progetto, deve essere conforme[4]. La modellazione dei requisiti avviene secondo le necessità degli utenti del sistema per risolvere le proprie problematiche.

2.2.1 Requisiti funzionali e non funzionali

I requisiti funzionali riguardano, attraverso l'implementazione delle varie funzionalità, il comportamento del sistema informatico nel suo complesso. Tra i requisiti funzionali rientrano anche i casi d'uso. Vengono definiti invece requisiti non funzionali tutti i requisiti che non sono determinanti nel funzionamento della soluzione informatica (Garzone), ma sono relativi alle sue proprietà nel suo complesso, come ad esempio prestazioni, scalabilità, usalibilità e prestazioni.

2.3 Casi d'uso

Un caso d'uso, seguendo le linee guida RUP (Rational Unified Process), è un insieme di istanze di casi d'uso, in cui ciascuna istanza è una sequenza di azioni che un sistema esegue per produrre un risultato osservabile e di valore per uno specifico attore[5]. Per convenzione, i casi d'uso che prevedono la parola chiave "gestisci" riuniscono gli obiettivi separati CRUD (create, retrieve, update, delete) in un unico caso CRUD.

2.3.1 Casi d'uso in Garzone

I casi d'uso possono essere descritti, in genere, in diversi modi. Per la definizione dei casi d'uso relativi a Garzone, si è optato per una stesura in formato breve, ovvero riepiloghi brevi e precisi relativi ai soli scenari di successo. Di seguito vengono elencati i principali casi d'uso.

Registra Negozio Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, compila manualmente i dati relativi alla sua registrazione. Il sistema, dopo la convalida dei dati e mediante l'interfaccia con il servizio di autenticazione degli utenti e di geolocalizzazione, registra l'utenza.

Registra Cliente Il cliente compila manualmente i dati relativi alla sua registrazione. Il sistema, dopo la convalida dei dati e mediante l'interfaccia con il servizio di autenticazione degli utenti e di geolocalizzazione, registra l'utenza.

Gestisci Vetrina Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, aggiunge o modifica prodotti e servizi del suo catalogo. L'utente in questione inserisce manualmente i dettagli del prodotto o servizio. Il sistema convalida i dati e registra l'inserimento o la modifica. I clienti saranno in grado di navigare sulla vetrina del negozio e visionarne il catalogo.

Gestisci Promozione Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, aggiunge una promozione relativa a un prodotto/servizio offerto. L'utente in questione inserisce manualmente i dati relativi alla promozione tra cui la data di scadenza. Il sistema convalida e registra la promozione. I clienti saranno in grado di visionare solo le promozioni attive del negozio ed eventualmente usufruirne.

Gestisci Appuntamento Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, crea un evento per un determinato orario referenziando un cliente coinvolto. L'addetto che registra l'appuntamento inserisce i dati manualmente e, dopo convalida e registrazione, il sistema notifica il cliente della creazione dell'evento.

Crea Ordine Il cliente, dopo essersi autenticato, dalla pagina di vetrina del negozio aggiunge prodotti messi a disposizione dal negozio al carrello. Il sistema dopo averne calcolato il totale richiede al cliente, tramite interfaccia grafica, l'inserimento manuale di dati relativi all'evasione dell'ordine. Nel caso in cui il cliente volesse che il suo ordine sia evaso ritirandolo in negozio non sarà necessario specificare un indirizzo di consegna, viceversa dovrà essere inserito. Il sistema, dopo la conferma da parte del cliente, registra l'ordine e il negozio viene notificato dell'avvenuta creazione.

Accetta Ordine Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, accetta la richiesta di ordine effettuata dal cliente. Dopo l'accettazione dell'ordine, il negozio procede con la sua evasione, specificando per gli ordini di consegna a domicilio un eventuale costo di consegna. Il sistema registra il cambio di stato dell'ordine. Il cliente dopo la modifica viene notificato automaticamente dell'avvenuto aggiornamento e del totale aggiornato, comprensivo del costo di consegna inserito.

Paga Ordine Il cliente, dopo l'autenticazione e dopo l'approvazione dell'ordine che prevede un pagamento online da parte del negozio e l'eventuale inserimento del costo

di consegna per gli ordini di consegna a domicilio, procede con il pagamento online. Il sistema calcola il totale dell'ordine comprensivo del costo di consegna e delega al sistema di autorizzazione dei pagamenti il completamento dell'operazione. Il sistema notifica il negozio e il cliente dell'avvenuto pagamento e aggiorna lo stato attuale dell'ordine.

Evadi Ordine Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, modifica lo stato dell'ordine. Per gli ordini con pagamento online il negozio procede con l'effettiva consegna e il cambio di stato dell'ordine. Per gli ordini che prevedono il ritiro in negozio la preparazione dei prodotti/servizi richiesti per il ritiro e il successivo cambio di stato dell'ordine. Il sistema registra il cambio di stato dell'ordine e notifica il cliente.

Invia Preventivo Il negozio, tramite il suo titolare o l'addetto all'utilizzo della piattaforma, inserisce manualmente i dati relativi a un preventivo. Il sistema, dopo aver convalidato i dati inseriti, invia il preventivo contenente i dettagli di un possibile ordine al cliente. Il cliente accetta il preventivo e il sistema aggiorna lo stato dell'ordine notificando il negozio dell'avvenuto aggiornamento.

Accetta Preventivo Il cliente, dopo la sua autenticazione e la ricezione di un preventivo da parte del negozio, accetta la proposta. Il sistema aggiorna lo stato dell'ordine e notifica il negozio dell'avvenuto cambiamento.

Invia Messaggio Il cliente, dopo la sua autenticazione, avvia una conversazione di chat con il negozio (o viceversa). Dopo la validazione del messaggio, il sistema provvede alla sua registrazione. Il negozio (il cliente) viene notificato della sua ricezione e viene abilitato ad inviare a sua volta un messaggio.

Abilita Negozio Il comune, dopo la sua autenticazione, abilita o disabilita un negozio, rendendone impossibile l'accesso e la visualizzazione sulla piattaforma.

Gestisci Comunicazione Il comune, dopo la sua autenticazione, aggiunge una comunicazione, la quale sarà visibile ai negozianti e agli utenti del marketplace.

Gestisci Comune L'amministratore di sistema, dopo la sua autenticazione, provvede a gestire i dati relativi ad un comune. Dopo la compilazione manuale dei dati del comune, il sistema li convalida e registra/aggiorna l'istanza.

Gestisci Negozio L'amministratore di sistema, dopo la sua autenticazione, provvede a gestire i dati relativi ad un negozio. Dopo la compilazione manuale dei dati del negozio, il sistema li convalida e registra/aggiorna l'istanza.

Gestisci Cliente L'amministratore di sistema, dopo la sua autenticazione, provvede a gestire i dati relativi ad un cliente. Dopo la compilazione manuale dei dati del cliente, il sistema li convalida e registra/aggiorna l'istanza.

Caso d'uso	**Attori coinvolti**
Registra Negozio	Negozio, Amministratore, Servizio autenticazione, Servizio di geolocalizzazione
Registra Cliente	Cliente, Servizio autenticazione, Servizio di geolocalizzazione
Gestisci Vetrina	Negozio, Amministratore
Gestisci Promozione	Negozio, Amministratore
Gestisci Appuntamento	Negozio, Cliente, Amministratore
Crea Ordine	Cliente
Accetta Ordine	Negozio, Amministratore
Paga Ordine	Cliente, Servizio autorizzazione pagamenti
Evadi Ordine	Negozio, Amministratore
Invia Preventivo	Negozio
Accetta Preventivo	Cliente
Invia Messaggio	Negozio, Cliente
Abilita Negozio	Comune, Amministratore
Gestisci Comune	Comune, Amministratore, Servizio autenticazione, Servizio di geolocalizzazione
Gestisci Negozio	Negozio, Amministratore, Servizio autenticazione, Servizio di geolocalizzazione
Gestisci Cliente	Cliente, Amministratore, Servizio autenticazione, Servizio di geolocalizzazione
Gestisci Comunicazione	Comune, Amministratore

Tabella 2.1: Riepilogo Casi d'uso e Attori coinvolti

2.4 Modellazione astratta e struttura del progetto

Garzone ha lo scopo di rendere possibile la transizione vera e propria di un negozio fisico al digitale fornendo ai negozianti una piattaforma gestionale efficiente e ai cittadini un efficacie canale di comunicazione. Per ogni attore è prevista la realizzazione di una propria piattaforma con un dominio web dedicato.
Tutte le piattaforme comunicano per la gestione dati con un sistema backend unico,

il quale a sua volta si interfaccia con sistemi per l'autorizzazione dei pagamenti, per l'autenticazione degli utenti e una base di dati.

2.4.1 Pannello Negozio

Il pannello del negoziante prevede l'implementazione delle seguenti funzionalità:

- **Registrazione**: consiste in una sezione dove viene richiesto al negoziante di registrarsi, selezionando prima il marketplace (comune) in cui è ubicato e poi inserendo manualmente sia dati personali che dati fiscali relativi al negozio fisico
- **Accesso**: prevede una sezione dove un negozio registrato effettua l'accesso previa una registrazione già effettuata e l'effettiva abilitazione da parte del comune
- **Dashboard**: contiene una data visualization tramite grafici dei dati più rilevanti riguardo la gestione del negozio
- **Gestione dati relativi all'utenza registrata**: prevede una sezione in cui è possibile per il negoziante modificare dati fiscali relativi al negozio o dati personali
- **Gestione funzionalità attive**: offre la possibilità al negoziante di abilitare/disabilitare funzionalità di Garzone relative al suo negozio, come ad esempio i pagamenti online
- **Gestione prodotti/servizi**: da questa sezione è possibile gestire mediante creazione, modifica ed eliminazione tutti i prodotti/servizi presenti nel catalogo del negozio fisico
- **Gestione promozioni**: da questa sezione è possibile inserire, modificare ed eliminare promozioni
- **Gestione appuntamenti**: da questa sezione è possibile inserire, modificare ed eliminare appuntamenti
- **Gestione ordini**: da questa sezione è possibile gestire gli ordini ricevuti dai clienti, i quali seguiranno un flusso di evasione prestabilito
- **Invio messaggi**: tramite questa sezione il negoziante potrà comunicare solo con clienti hanno precedentemente inviato un messaggio al negozio
- **Visualizzazione comunicazioni**: contiene le comunicazioni create dal comune in cui il negozio è registrato

2.4.2 Pannello Comune

Il pannello del comune prevede l'implementazione delle seguenti funzionalità:

- **Accesso**: prevede una sezione dove un comune registrato effettua l'accesso, previa una registrazione già effettuata dall'amministratore di Garzone e l'effettiva abilitazione
- **Dashboard**: contiene una data visualization tramite grafici dei dati più rilevanti riguardo la gestione del negozio
- **Gestione dati relativi all'utenza registrata**: prevede una sezione in cui è possibile per il comune modificare dati fiscali relativi all'organo amminsitrativo e dati relativi al suo referente
- **Abiltazione negozi**: da questa sezione è possibile abilitare o disabilitare i negozi del proprio comune (marketplace)
- **Gestione comunicazioni**: tramite questa sezione il comune potrà comunicare ai negozianti del proprio marketplace avvisi rilevanti

2.4.3 Pannello Amministratore

Il pannello di amministrazione prevede l'implementazione delle seguenti funzionalità:

- **Accesso**: prevede una sezione dove l'amministratore può accedere alla sua piattaforma
- **Dashboard**: contiene una data visualization tramite grafici dei dati più rilevanti riguardo negozi, comuni e clienti
- **Gestione comuni**: tramite questa sezione l'amministratore può gestire, inserendo, modificando o eliminando, i dati dei comuni
- **Gestione negozi**: da questa sezione è possibile gestire tramite inserimento, modifica ed eliminazione, i dati fiscali, i dati relativi al negozio fisico, le chat, gli ordini e gli appuntamenti dei negozianti nei vari marketplaces
- **Gestione clienti**: tramite questa sezione l'amministratore può gestire, inserendo, modificando o eliminando, i dati dei clienti e relativi ordini, chat e appuntamenti
- **Gestione promozioni**: tramite questa sezione l'amministratore può gestire, inserendo, modificando o eliminando, i dati delle promozioni create dai negozianti

2.4.4 Pannello Cliente

Il pannello destinato ai clienti, invece prevede lo sviluppo delle seguenti funzionalità:

- **Registrazione**: consiste in una sezione dove viene richiesto al cliente di registrarsi, inserendo manualmente i suoi dati personali
- **Accesso**: prevede una sezione dove il cliente può accedere alla piattaforma di Garzone
- **Visualizzazione marketplace**: prevede la possibilità per il cliente di visualizzare tutti i negozi, registrati e abilitati dalla piattaforma, di un comune da lui selezionato
- **Visualizzazione promozioni**: revede la possibilità per il cliente di visualizzare tutte le promozioni inserite dai negozi e attualmente attive (non scadute) per il marketplace selezionato
- **Visualizzazione negozio**: prevede la possibilità per il cliente di visualizzare i dettagli di registrazione, il catalogo prodotti/servizi con la possibilità di aggiunta al carrello per un successivo ordine, le promozioni e gli slot di appuntamento già occupati di un negozio specifico selezionato dal marketplace
- **Invio messaggi**: tramite questa sezione il cliente, previa registrazione e accesso, ha la possibilità di inviare messaggi ad un negozio selezionato
- **Richiesta preventivo**: tramite questa funzionalità un cliente può richiedere un preventivo al negoziante indicando le sue necessità
- **Gestione ordini**: da questa sezione è possibile gestire gli ordini inviati ai negozi, i quali seguiranno un flusso di evasione prestabilito
- **Gestione appuntamenti**: da questa sezione è possibile gestire, modificando o annullando, gli appuntamenti presi con i negozi dei vari marketplaces
- **Visualizzazione comunicazioni**: contiene le comunicazioni create dal comune (marketplace) selezionato

I domini della soluzione costituiscono una struttura gerarchica ad albero, in cui il pannello cliente è consultabile tramite dominio principale, mentre gli altri pannelli tramite sottodomini. L'idea generale segue questa struttura:

Figura 2.1: Struttura domini e sottodomini Garzone

Le varie piattaforme comunicano tra loro interfacciandosi al servizio di autenticazione delle utenze e ad un sistema backend condiviso, il quale a sua volta si interfaccia a un'apposita base di dati e al servizio di autorizzazione dei pagamenti.

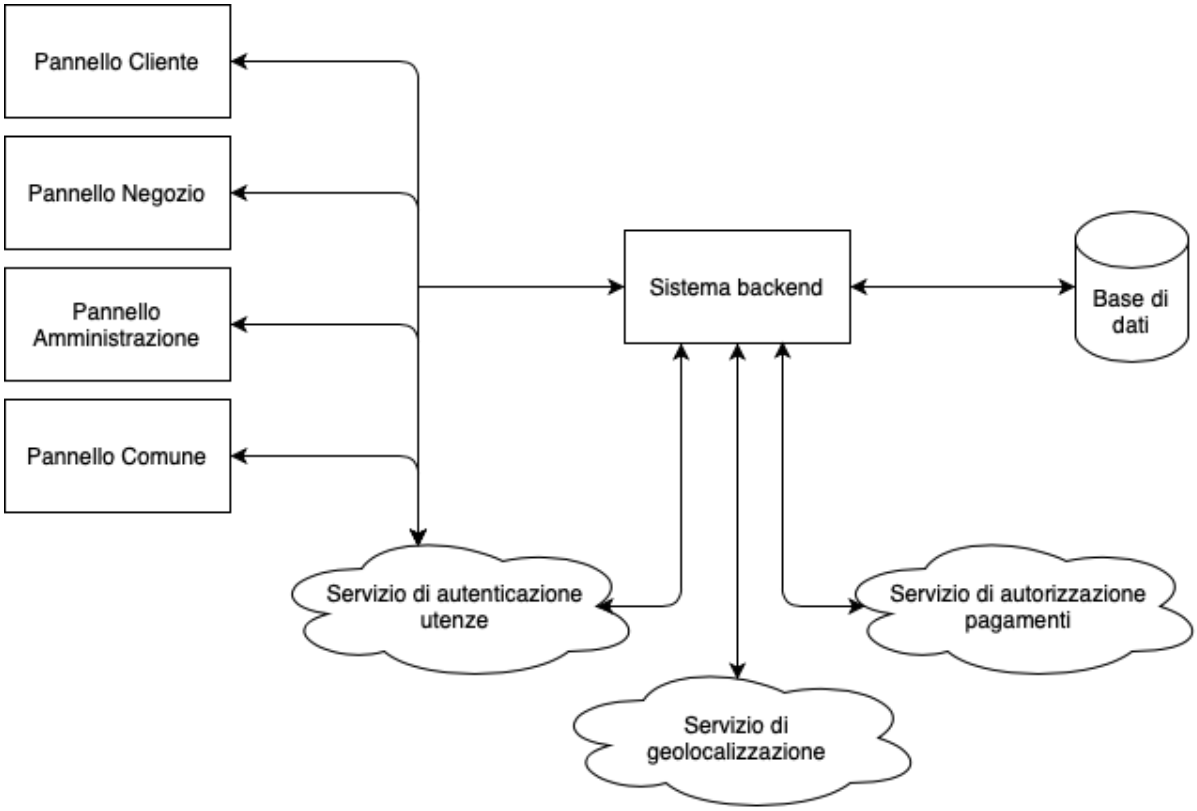


Figura 2.2: Modello complessivo astratto di implementazione

Capitolo 3

Componente tecnologica

3.1 Introduzione

La realizzazione del progetto prevede, dopo la definizione di una precisa architettura, l'utilizzo di diverse componenti tecnologiche, come ad esempio l'utilizzo di un framework, per soddisfare i vari requisiti utente.

3.2 Strumenti utilizzati per l'organizzazione del lavoro

3.2.1 ClickUp

Ai fini di organizzare per il meglio il lavoro da svolgere, l'utilizzo della piattaforma ClickUp ha permesso al team di sviluppo la suddivisione dei compiti da svolgere mediante la definizione di macro-task, organizzati in dashboard visibili dal settore amministrativo e tecnico. Un macro-task rappresenta l'implementazione di un caso d'uso utente e corrisponde ad un rilascio in produzione. Inoltre prevede un'univocità all'interno della dashboard ed è visibile e modificabile dal reparto di amministrazione per supervisionare l'andamento del progetto. A sua volta un macro-task è suddiviso in task, i quali sono organizzati nei diversi pannelli (admin, negozio, utente, comune) riprendendo il codice univoco a cui è associato il macro-task. Ogni task è inserito nella dashboard gestita dal team di sviluppo e corrisponde all'implementazione di una micro-funzionalità. Deve soddisfare precisi requisiti, di seguito elencati.

- **Durata Limitata**: un task deve prevedere un tempo di implementazione da una alle quattro ore complessive
- **Atomicità**: un task deve prevedere l'implementazione/correzione di funzionalità singole

- **Univocità**: un task non deve essere mai ripetuto, per implementare bugfixes è necessario creare un nuovo macro-task e di seguito organizzarne la suddivisione
- **Reperibilità**: un task deve essere facilmente trovato ricercando parole chiave definite nel titolo, il quale segue lo standard «*COD # - Titolo breve*»

Ogni task nel flusso di lavoro può assumere diversi status:

- **Open**: il task è appena stato creato e non è ancora stato implementato
- **In Progress**: il task è in corso di implementazione da parte del team di sviluppo
- **Review**: il task è in corso di revisione da parte dei preposti al testing
- **Closed**: il task è concluso e testato

Un macro-task può passare nello status "closed" solo quando tutti i task a lui relativi sono in closed. A questo punto è possibile procedere con il rilascio della piattaforma in produzione.

Figura 3.1: Esempio di organizzazione macrotask - tasks di una funzionalità di Garzone

3.2.2 Bitbucket

Per consentire la condivisione del codice sorgente tra il team di sviluppo è stato previsto l'utilizzo della piattaforma Bitbucket, un servizio di hosting per progetti implementati tramite utilizzo del protocollo di version control Git. Per l'implemntazione del progetto sono state create quattro apposite repository:

- **garzone-serverless**: la quale comprende tutto il codice sorgente relativo alla componente backend
- **garzone-webapp-admin**: relativa al pannello admin
- **garzone-webapp-negozio**: relativa al pannello negozio
- **garzone-webapp-comune**: relativa al pannello comune
- **garzone-webapp-utente**: relativa al pannello utente

Le operazioni di version control sono state effettuate tramite l'utilizzo di Github Desktop, un'applicazione che utilizza il protocollo Git mediante un'interfaccia grafica. Ad ogni task del progetto implementato è correlato un commit creando quindi un riferimento al flusso di lavoro definito in partenza.

3.2.3 Separazione ambienti di lavoro

La separazione degli ambienti di lavoro è necessaria per permettere al team di sviluppo di lavorare alle implementazioni di test parallelamente a quelle in produzione. In questo modo si consente un corretto utilizzo della piattaforma da parte dell'utenza su una versione testata e la possibilità da parte degli sviluppatori di procedere con il live deploy solo dopo aver testato il deploy di test. Al progetto vero e proprio Firebase-hosted ne è stato quindi affiancato un altro parallelo, il quale presenta lo stesso identificativo *(piattaforma-garzone)* con l'aggiunta del suffisso *"-test"*. I due progetti presentano la medesima suddivisione di sottodomini e si interfacciano a backend differenti, i quali a loro volta si interfacciano a due istanze di basi di dati differenti. Per l'avvio e la sua messa in produzione, il progetto prevede l'utilizzo di script appositi per ogni pannello, ognuno dei quali ha una funzione specifica di avvio o di deploy. Di seguito vengono proposti gli script relativi alla parte di backend e relativi alla componente frontend del pannello utente, seguiti da una panoramica generale dei servizi a cui si interfacciano le varie piattaforme.

Pannello Utente

yarn start:dev	• Autenticazione su *piattaforma-garzone-test* • Funziona nella directory *garzone-webapp-utente*
yarn start:prod	• Autenticazione su *piattaforma-garzone* • Funziona nella directory *garzone-webapp-utente*
yarn deploy:dev	• Pubblica l'applicazione nel progetto Firebase *piattaforma-garzone-test* nel multi-site di default • Funziona nella directory *garzone-webapp-utente*
yarn deploy:prod	• Pubblica l'applicazione nel progetto Firebase *piattaforma-garzone* nel multi-site di default • Funziona nella directory *garzone-webapp-utente*

Figura 3.2: Documentazione degli script di avvio/deploy del pannello utente

Garzone Serverless

npm run-script serve:dev	• Avvia l'emulatore in locale collegandosi mediante proxy all'istanza db **DEV** • Funziona nella directory *garzone-serverless/**functions***
npm run-script serve:prod	• Avvia l'emulatore in locale collegandosi mediante proxy all'istanza db **PROD** • Funziona nella directory *garzone-serverless/**functions***
npm run-script deploy:dev	• Pubblica le functions nel progetto Firebase *piattaforma-garzone-test* • Funziona nella directory *garzone-serverless/**functions***
npm run-script deploy:prod	• Pubblica le functions nel progetto Firebase *piattaforma-garzone* • Funziona nella directory *garzone-serverless/**functions***

Figura 3.3: Documentazione degli script di avvio/deploy della componente backend (Firebase Cloud Functions)

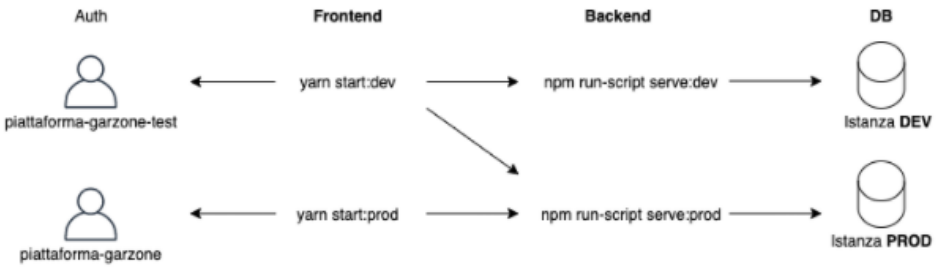


Figura 3.4: Sommario script di avvio dei progetti di test e live e relative interfacce

3.3 Javascript e NodeJS

Il progetto è interamente realizzato tramite l'utilizzo del linguaggio Javascript. Fu introdotto nel 1995 come strumento per inserire programmi nelle pagine web del browser Netscape Navigator. Successivamente è stato via via adottato da tutti gli altri browser grafici, rendendo realizzabili moderne applicazioni Web con le quali si può interagire direttamente, senza dover necessariamente ricaricare la pagina. Viene anche utilizzato in siti più tradizionali per offrire varie forme di interattività.[6]. Node.js è stato creato tramite l'utilizzo di Javascript. In particolare si tratta di un ambiente asincrono ed event driven per la costruzione di applicazioni distribuite basate su Javascript. Node.js non è un framework, lavora infatti a livelli più bassi, è sostanzialmente un interprete con alcune librerie di base non native del linguaggio Javascript, ad esempio per l'I/O su file, per l'implementazione di server per vari protocolli standard. Sfrutta il loop di gestione degli eventi di Javascript, in particolare sulla possibilità di definire funzioni callback per implementare uno schema di I/O non bloccante.

Linee guida stesura del codice Il team di sviluppo per la stesura del codice ha adottato uno standard comune a tutte le sue componenti. La struttura dei file di codice sorgente è stati quindi organizzata in:

- **Screens**: ogni schermata dell'applicazione è rappresentata da un file di codice sorgente in questa directory
- **Components**: ogni componente complesso all'interno di una schermata deve essere inserito in un file di codice sorgente apposito, per consentire una maggiore leggibilità e manutenibilità della soluzione
- **Controllers**: ogni componente dell'UI per interfacciarsi con la logica dell'applicativo comunica con un controller che ha lo scopo di delegare questo compito alla parte di backend
- **Assets**: ogni componente grafica o file di dati statico è organizzato in questa cartella e utilizzato in vari punti del codice
- **Config**: i file di configurazione delle librerie esterne, la validazione dei dati e impostazioni varie, quali palette di colori e stringhe statiche, sono organizzati in questa cartella
- **Utility**: tutte le funzioni delegate a compiti complessi, come il calcolo del totale dei prezzi degli articoli di un carrello, sono contenute in file in questa directory

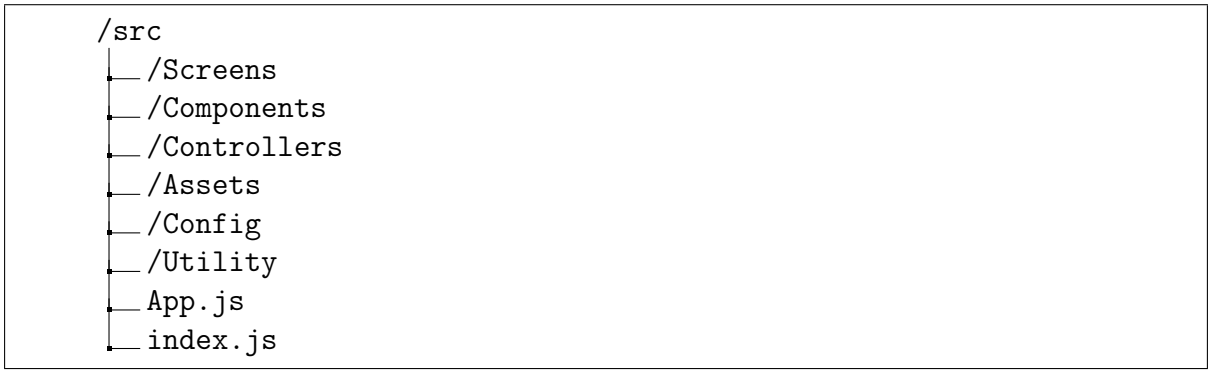


Figura 3.5: Struttura delle directory e dei file di un pannello di Garzone

- *App.js*: rappresenta il codice sorgente radice dell'interfaccia grafica dell'applicazione react
- *index.js*: rappresenta il codice sorgente radice dell'applicazione stessa e ha il compito di eseguire l'inizializzazione delle componenti logiche e grafiche, come lo Store di Redux, una libreria di React per la gestione dello stato globale

Inoltre è stato previsto uno stile di stesura ispirato a Airbnb Javascript [7], che contiene un elenco di best practices per evitare errori di scrittura e con lo scopo di coordinare il metodo di stesura di codice del team di sviluppo.

3.3.1 npm e Yarn

`npm` è un sistema di gestione pacchetti scritti in Javascript in ambiente Node.js. Di norma contengono collezioni di funzioni, alcune delle quali esportate, ovvero "visibili" all'esterno del modulo e quindi richiamabili dal progetto. Per la realizzazione di Garzone sono stati utilizzati numerosi pacchetti, con il compito di interfacciarsi a API esterne e realizzare componenti grafiche complesse, come ad esempio mappe interattive. Inoltre è stato utilizzato per l'implementazione di script di esecuzione e rilascio dell'applicativo con configurazioni ad hoc. Per le componenti frontend è stato invece utilizzato `Yarn`, un packet manager costruito a un livello di astrazione superiore di `npm`, fornendone una semplificazione della sintassi delle istruzioni eseguibili.

3.4 Componente Frontend

3.4.1 Linee guida grafiche per l'interfaccia utente

Per la realizzazione delle interfacce grafiche il team di sviluppo ha fatto uso di mockups realizzati dal team grafico. Ogni mockup prevede file generati dal software `AdobeXD` dai quali è possibile esportare ogni componente grafico (asset) per poi importarlo nel progetto. La progettazione ha previsto l'implementazione di standard condivisi tra i pannelli, ad esempio una palette di colori, il font utilizzato dalla Pubblica Amministrazione su indicazione delle linee guida AGID, le dimensioni di vari componenti e del font, animazioni grafiche e così via. Ogni schermata dell'applicativo prevede quindi una controparte realizzata in `AdobeXD`.

3.4.2 ReactJS

Il framework su cui si basano le intere interfacce grafiche di Garzone è `ReactJS`, una libreria Javascript che mette a disposizione numerose funzionalità logiche e grafiche per ottenere una user experience efficace. L'utilizzo di `ReactJS` ha messo a disposizione del team di sviluppo numerosi vantaggi, tra cui:

Riutilizzo del codice `ReactJS` è una libreria **component-based**, ovvero permette l'incapsulamento di componenti logico-grafici per l'utilizzo in varie parti dell'applicativo attraverso l'utilizzo delle `props`, parametri utilizzabili da un componente "figlio" forniti dal componente "genitore".

Manutenibilità del codice L'incapsulamento dei componenti prevede, tra le altre cose, una maggiore facilità nel manutenere il codice. Infatti è possibile, tramite una singola modifica, modificare lo stesso componente in tutte le parti dell'applicativo.

State management e Virtual DOM `ReactJS` prevede che al cambiare delle informazioni (dati) contenute all'interno dello stato di un componente, l'interfaccia grafica venga aggiornata automaticamente con le nuove informazioni contenute nello stato. Questo è reso possibile dall'utilizzo del Virtual DOM, ovvero una rappresentazione del DOM originale della pagina. Al verificarsi di un evento, React occorre le modifiche necessarie sul Virtual DOM e successivamente, dopo l'analisi delle differenze tra Virtual DOM e DOM effettivo, riporta le modifiche sul DOM vero e proprio. Il Virtual DOM è composto quindi da nodi gerarchici chiamati `ReactNode`, dai quali ereditano anche gli elementi `ReactElement`, l'elemento più comune in React utilizzato per il rendering di componenti

scritti sfruttando il JSX, e `ReactText`, l'oggetto per il rendering del testo nei nodi del virtual DOM.

I componenti in React seguono un ciclo di vita predefinito costituito dai principali eventi *init*, *render* e *mount*.

Inizializzazione L'inizio del ciclo di vita di un componente React prevede la definizione del suo stato iniziale, ovvero un oggetto Javascript utilizzato da React per la creazione del Virtual DOM. L'oggetto *«state»* creato sarà quindi accessibile dal componente solo dopo la sua inizializzazione.

Rendering React nella fase di rendring ha il compito di materializzare i nodi contenuti nel componente, facendo uso dello stato creato nella fase di inizializzazione. Il metodo di render viene richiamato ogni qualvolta l'oggetto di stato viene aggiornato ed è richiesto per la definizione di un componente.

Montaggio e Smontaggio React prevede la possibilità di intercettare la fase di ciclo di vita in cui il componente viene "montato" sulla pagina, ovvero quando la libreria ha concluso il rendering di tutti i nodi contenuti nel componente. Così come esiste la possibilità di intercettare il montaggio, è possibile intercettare anche l'evento di smontaggio del componente, ovvero quando gli elementi del componente facenti parte del DOM vengono rimossi. Di norma per un componente ad una navigazione verso una schermata dell'applicazione esistono una e una sola fase di montaggio e una e una sola fase di smontaggio.

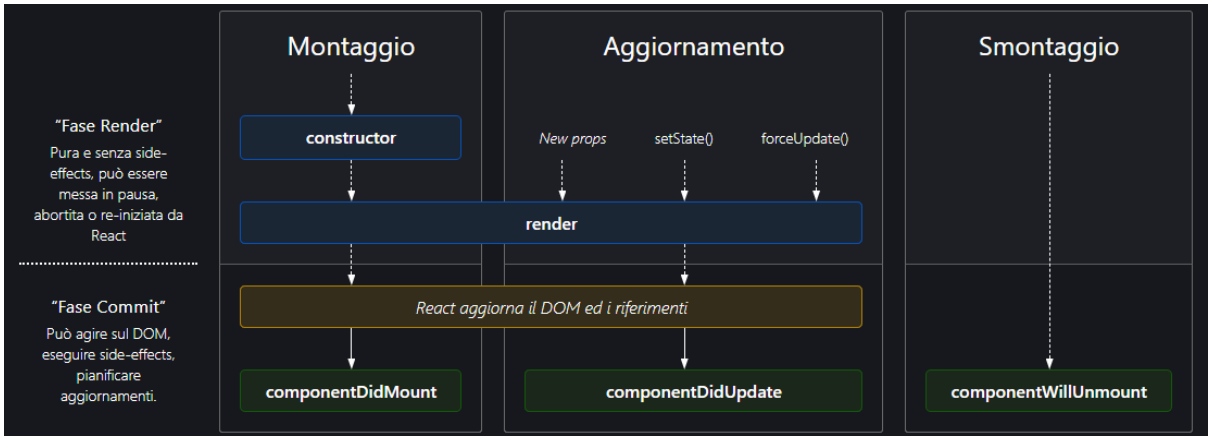


Figura 3.6: Ciclo di vita di un React Component [8]

In React è possbile implementare due tipologie di componenti:

- **Class Components**: estensioni della classe `React.Component`
- **Function Components**: componenti funzionali basati sulla logica degli *«hooks»*, ovvero funzionalità che permettono il riutilizzo dello stato dei componenti nelle varie gerarchie, senza modificarne la logica e permettono una maggiore facilità d'uso riguardo il cilo di vita di un componente

Figura 3.7: Esempio di Function Component in Garzone

Figura 3.8: Esempio di Class Component in Garzone

3.4.3 Gestione dello stato globale: Redux

In React la gestione dello stato è possibile nativamente all'interno di ogni singolo componente, ma così facendo emergono diverse problematiche come l'interoperabilità delle operazioni tra gli stati dei vari componenti e la presenza di pattern non standard per la loro gestione. Redux viene introdotto per risolvere, tra le altre cose, queste problematiche, prevedendo un flusso comune a tutte le operazioni di gestione dello stato:

1. Definizione di una `Action` che prevede un tipo (`type`) e un set di dati (`payload`)
2. Tramite un `Dispatcher` l'`Action` viene eseguita sullo `Store`
3. Un `Reducer` aggiorna lo `Store` dell'applicazione tramite l'analisi del parametro `type` dell'`Action` sostituendo l'attuale stato con i dati contenuti nel `payload`

Store Lo `Store` in Redux è l'entità che contiene lo stato globale dell'applicazione

Reducer Un `Reducer` è una funzione che ha lo scopo di aggiornare lo stato attuale dell'applicazione in un nuovo stato

Action Un `Action` è un oggetto Javascript che per convenzione presenta un parametro `type`, utilizzato dal `Reducer` per interpretare come cambio di stato, e un parametro `payload` che contiene i dati utilizzati dal `Reducer` per aggiornare lo stato

Dispatcher Il `Dispatcher` ha il compito di inviare allo `Store` un `Action`, la quale, tramite il `Reducer`, esegue l'aggiornamento di stato definito nel `type` dell'`Action`

In Garzone Redux è utilizzato per la gestione della fase di login, salvando globalmente i dati dell'utente, e per la gestione dell'interfaccia grafica di chat e ordini, prevedendo ad esempio cambi di stato per l'aggiornamento realtime dei badge delle chat non lette o dei nuovi ordini.

3.4.4 antd

Per facilitare l'implementazione dei componenti grafici è stato previsto l'utilizzo del framework grafico *antd*. Creato come libreria UI di React, antd contiene un set di componenti e demo per costruire interfacce grafiche interattive[9]. In Garzone, antd è stato importato tramite l'utilizzo del packet manager `Yarn`, e ne sono stati implementati i vari moduli facendo uso del tree shaking introdotto in Javascript ES6 con l'istruzione `import`.

Sovrascrizione stili con Less Antd offre l'opportunità di sovrascrivere lo stile dei propri componenti utilizzando Less, una libreria che permette di scrivere stili con una sintassi formale e una logica definita. In Garzone sono state sovrascritte gran parte delle variabili di stile implementate da antd e sono stati opportunamente modificati gli script di esecuzione della soluzione, inserendo e utilizzando `craco` *(Create React App Configuration Override)*: un layer sovrastante il packet manager che offre la possibilità di aggiungere configurazioni particolari alla soluzione.

3.5 Componente Backend

3.5.1 Google Cloud

La realizzazione della componente backend ha previsto un'interfaccia all'ecosistema Google Cloud che, tra le altre cose, mette a disposizione un ambiente di hosting, un ambiente FaaS (Function as a Service), un servizio completamente gestito per database relazionali e altri tool di sviluppo e documentazione come Google Workspace.
Per la realizzazione di Garzone sono stati creati due progetti Google Cloud separati

- **garzone-test**: per l'ambiente di test
- **garzone**: per l'ambiente di produzione

ognuno dei quali si interfaccia a micro-servizi indipendenti sia interni che esterni all'ambiente Google Cloud. Ogni componente del team di sviluppo ha un'utenza collegata al progetto con i privilegi impostati sulle mansioni affidate per la realizzazione del progetto e sono previste stringenti regole di sicurezza per monitorare gli accessi a credenziali e pannelli di amminsitrazione.

Cloud SQL

Cloud SQL, è un servizio completamente gestito per l'hosting di database relazionali come mySql, PostgreSQL e SQL Server. A differenza di soluzioni in-house, offre costi di manutenzione contenuti, continuità aziendale, scalabilità in base al consumo e regole di sicurezza e conformità sempre aggiornate. In Garzone la connessione alla base di dati è prevista solo dalla componente FaaS in due modi:

- **Localmente**: la connessione al servizio avviene dalla sorgente FaaS mediante l'interfaccia dell'emulatore in locale alla base di dati tramite un auth proxy fornito da Google Cloud in fase di setup dell'ambiente

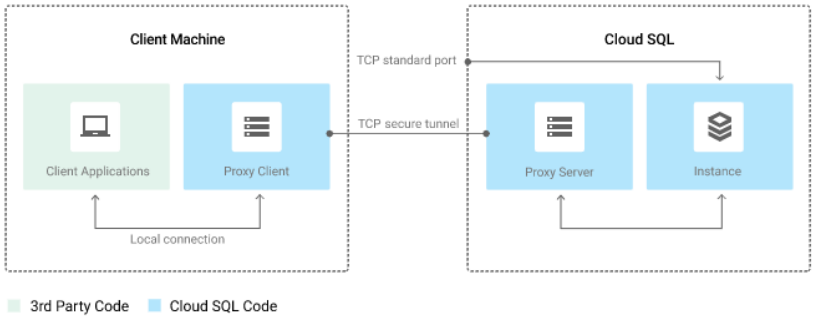


Figura 3.9: Connessione da emulatore FaaS a isitanza CloudSQL

- **Da remoto**: la connessione al servizio avviene dalla sorgente FaaS presente sull'host remoto

La configurazione della connessione è estratta dalle variabili d'ambiente impostate nel progetto di backend e prevede i seguenti parametri:

- *Host*: `localhost` (solo locale), rappresenta l'host a cui effettuare il collegamento al servizio di CloudSQL
- *Port*: `3306` (solo locale), rappresenta la porta del collegamento
- *User*: rappresenta l'utenza per l'autenticazione della connessione all'istanza
- *Password*: rappresenta la password per l'autenticazione della connessione all'istanza
- *Database*: rappresenta il nome del database a cui collegarsi
- *Socket Path*: rappresenta il path dell'istanza

Per un'ulteriore gestione della base di dati è stato utilizzato il software MySql Workbench, il quale fornisce un'interfaccia grafica interattiva per eseguire operazioni su database.

3.5.2 Firebase

Firebase è un servizio BaaS (Backend-as-a-service) che consente lo sviluppo di applicazioni offrendo la possibilità di implementare numerose funzioni. I servizi offerti si suddividono in tre macro-categorie[10]:

- **Build**
  - **Realtime Database**: utilizzato per le funzionalità di gestione ordini e chat realtime
  - **Authentication**: utilizzata per la fase di login, registrazione e gestione degli utenti
  - **Hosting**: utilizzato per il servizio hosting delle piattaforme web
  - **Cloud Storage**: utilizzato per l'upload di immagini da parte dell'utenza
  - **Cloud Functions**: utilizzate per la parte logica degli applicativi e interfacciarsi alla base di dati
- **Release and Monitor**
  - **Performance e Monitoring**: utilizzati per la gestione degli errori

- **Engage**
  - **Analytics**: servizio utilizzato per visionare il traffico e azioni dell'utenza per poi tracciare strategie di engagement

Firebase offre una console di amministrazione che riprende alcune delle funzionalità offerte da quella di Google Cloud, semplificandone l'interpretazione e l'utilizzo. L'ecosistema Google permette quindi di interfacciarsi direttamente alla soluzione sia da Firebase che dalla piattaforma Google Cloud.

Firebase Authentication

La gestione delle utenze in Garzone è delegata a Firebase Authentication, un servizio della suite Google che mette a disposizione un SDK utilizzabile sia dalla componente frontend, sia dalla componente backend.

Firebase Functions

La componente backend dell'applicativo è realizzata tramite l'implementazione delle Firebase Functions, endpoint creati utilizzando un framework serverless che eseguono codice in risposta ad eventi scatenati da altri prodotti Firebase o eventi HTTP. Il ciclo di vita di una cloud function prevede:

1. Stesura del codice e configurazione dei parametri di esecuzione
2. Deploy (rilascio) della funzione
3. Trigger dell'evento con invocazione ed esecuzione del codice
4. Se la funzione sta già gestendo numerosi eventi, Firebase crea altre istanze della function velocizzando l'esecuzione complessiva della richiesta

In Garzone il codice sorgente delle Firebase Functions presenta una suddivisione in base ai claim dell'utenza. Ad ogni file di functions corrisponde un attore dell'applicativo.

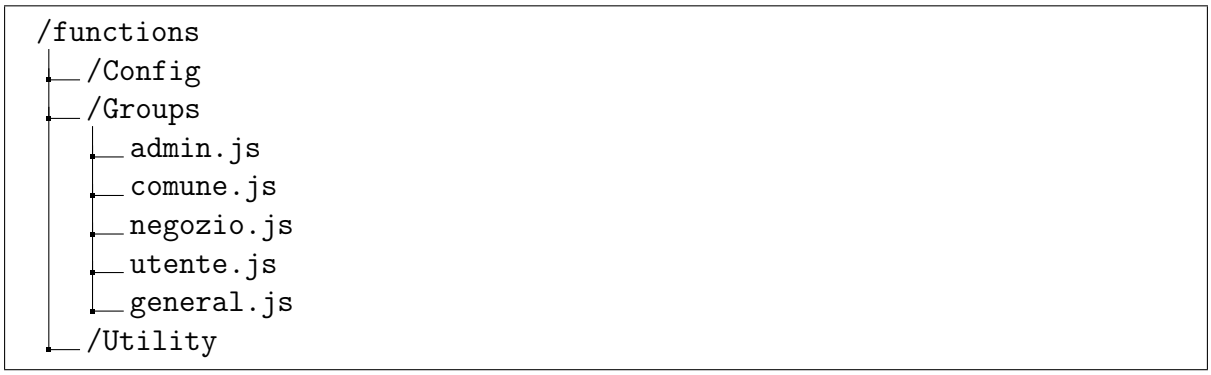


Figura 3.10: Struttura delle directory e dei file delle Firebase Functions di Garzone

Ogni Firebase Function prevede una parte di autenticazione tramite l'utilizzo del *context*, un parametro messo a disposizione in ogni funzione che fornisce informazioni riguardo la provenienza della richiesta. Inoltre prevede una parte di validazione dei dati proveniente dalla richiesta e infine l'esecuzione di codice in base alla funzionalità richiesta. Di seguito un esempio dell'implementazione di una Firebase Function per l'aggiornamento dei campi di un record su database.

```
1    exports.aggiornaModuloOrdini = functions
2  .region("europe-west1")
3  .https.onCall(async (data, context) => {
4    if (!policies.isProprietario(context, data.id)) {
5      throw new functions.https.HttpsError("permission-denied", "
     Accesso negato");
6    } else {
7      const aggiornaModuloOrdiniSchema = Yup.object({
8        id: Yup.number().required("Id mancante"),
9        modulo_ordini: Yup.boolean(),
10       ritiro: Yup.boolean(),
11       consegna: Yup.boolean(),
12       dettagli_consegna: Yup.string()
13         .nullable()
14         .strict(true)
15         .trim("Dettagli consegna con spazi vuoti iniziali/finali")
16         .max(1500, "Dettagli consegna > 1500 caratteri"),
17     });
18
19     //Validazione richiesta
```

```
20      const { error, value } = aggiornaModuloOrdiniSchema.validate(data
    );
21      if (error) {
22        throw new functions.https.HttpsError("invalid-argument", error,
     {
23          dettaglio: "Errore nella validazione dei campi",
24        });
25      }
26
27      //Destrutturazione richiesta
28      const {
29        id,
30        modulo_ordini,
31        ritiro,
32        consegna,
33        dettagli_consegna,
34      } = data;
35
36      //Composizione dati da inserire [campo_db: valore]
37      const negozioRecord = {
38        modulo_ordini: modulo_ordini,
39        ritiro: ritiro,
40        consegna: consegna,
41        dettagli_consegna: dettagli_consegna,
42      };
43
44      //Update del record con i dati della richiesta
45      try {
46        const pool = await db.connection;
47        await pool.query(`UPDATE negozi SET ? WHERE id=?`, [
    negozioRecord, id]);
48      } catch (error) {
49        throw new functions.https.HttpsError("failed-precondition",
    error, {
50          dettaglio: "Errore nell'aggiornamento del record su DB",
51        });
52      }
53
54      return { success: true };
55    }
56  });
```

Realtime Database

Firebase Realtime Database è un database NoSQL hostato in cloud che permette il salvataggio e la sincronizzazione dei dati condivisi tra gli utenti in tempo reale. In Garzone è stato implementato per la gestione dei servizi di chat e gestione ordini.

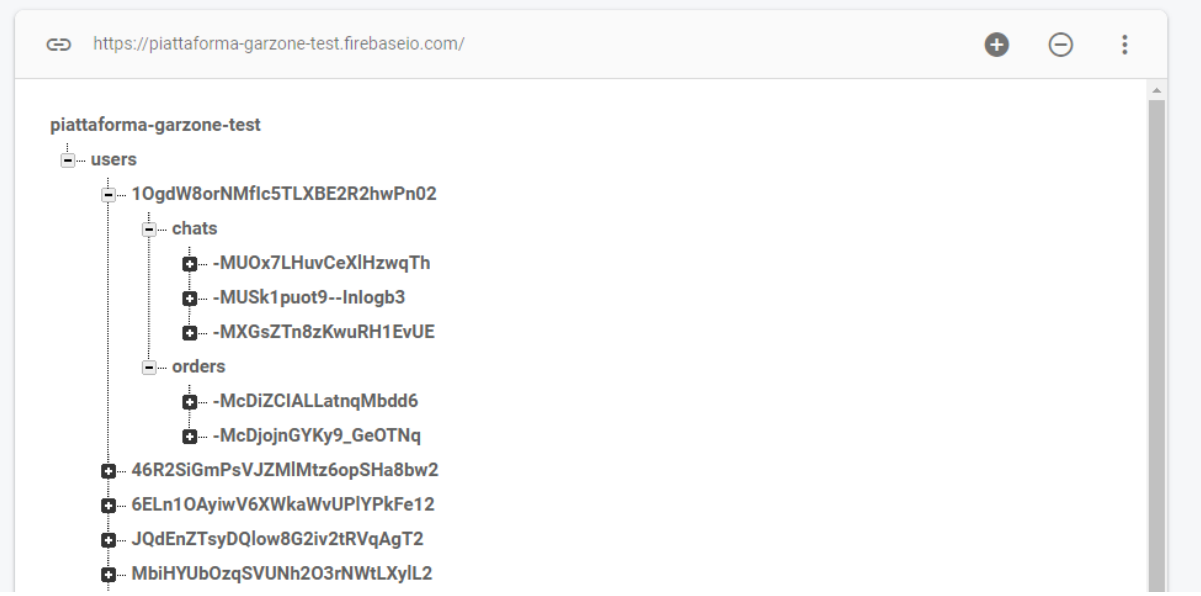


Figura 3.11: Porzione di Firebase Realtime Database in Garzone

Firebase Authentication

La registrazione, l'accesso e la gestione delle utenze in Garzone sono delegate al servizio di Firebase Authentication, il quale mette a disposizione l'implementazione del suo SDK. Le librerie su cui si basa il servizio sono OAuth 2.0 e OpenID Connect, inoltre Firebase Authentication fornisce servizi di autenticazione di terze parti come Google, Facebook, Twitter e Github, previsti come funzionalità aggiuntive per sviluppi futuri di Garzone. Per l'applicativo è stata inizialmente prevista un'autenticazione tramite mail e password insieme ai servizi di reset.

Firebase Cloud Storage

La gestione delle immagini è affidata al servizio di Firebase Cloud Storage, il quale presenta numerosi vantaggi, tra cui:

- Operazioni di download e upload indipendenti dalla qualità di rete del client che le esegue
- Modello di sicurezza per consentire l'accesso ai file solo a utenze autorizzate

- Scalabilità

Ogni file caricato sulla piattaforma viene depositato nel file system di Google Cloud, chiamato bucket. In Garzone il bucket presenta tre directory:

- /**categorie**, che contiene le immagini delle categorie merceologiche dei negozi, utilizzabili quando un negozio non fornisce un immagini profilo
- /**comuni**, contenente i loghi dei comuni iscritti a Garzone
- /**negozi**, contenente i loghi e le immagini relative a prodotti, servizi e promozioni di ogni negozio

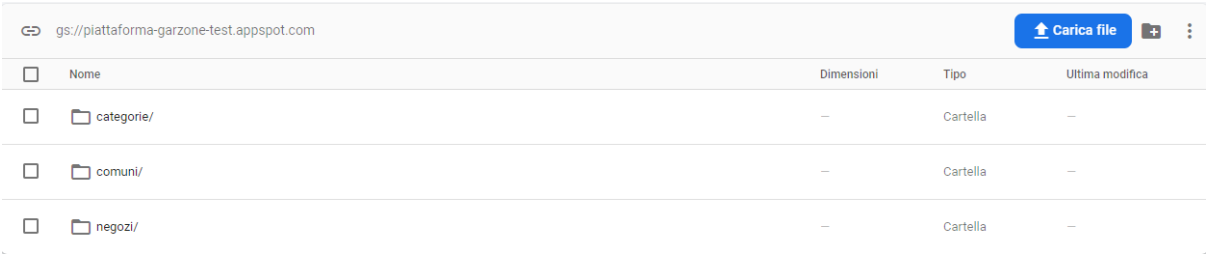


Figura 3.12: Interfaccia grafica console Firebase del Bucket di Garzone

3.5.3 AWS

AWS (Amazon Web Services) è un fornitore di servizi web IaaS/PaaS utilizzabili dall'utenza con lo scopo, tra le altre cose, di implementare funzionalità aggiuntive alle proprie soluzioni.

Simple Email Service

In Garzone per lo scambio di e-mail transazionali è stato previsto l'utilizzo del servizio web Simple Email Service. Le opzioni flessibili di distribuzione degli IP e di autenticazione delle e-mail offerte da Amazon SES consentono di incrementare l'efficacia del recapito e di proteggere la reputazione del mittente, mentre le analisi di invio misurano l'impatto di ciascuna e-mail[11]. Le funzionalità di SES sono state implementate utilizzandone l'SDK nella componente di backend, installandolo tramite il package manager `Yarn`.

3.5.4 Stripe

Stripe è un servizio web che fornisce prodotti per ricevere pagamenti. In Garzone è previsto l'utilizzo del servizio Stripe *«Connect»*, dedicato ai marketplace, ovvero piattaforme intermediarie di vendita tra venditore e consumatore. Stripe fornisce un SDK installabile

tramite packet manager `Yarn`, il quale offre la possibilità di implementare le funzioni per ricevere pagamenti, gestire i payout (pagamenti ai venditori), gestire storni e rimborsi, prevenire frodi e inviare ricevute.

Stripe Connect

Connect è un set di strumenti e API programmabili che semplifica i pagamenti sulla tua piattaforma software, la creazione di marketplace e il pagamento di venditori o fornitori di servizi a livello globale, il tutto delegando a Stripe la gestione della conformità dei pagamenti[12]. Prevede diverse funzionalità necessarie per il funzionamento di un marketplace, come Garzone

- Attivazione e verifica delle utenze (negozi)
- Accettazione dei pagamenti
- Invio di pagamenti agli utenti (negozi)
- Gestione delle transazioni e sicurezza

Capitolo 4

Funzionalità e relative implementazioni

4.1 Gestione utenze

Le utenze in Garzone sono rappresentate da singoli record sulla base di dati a cui sono referenziate istanze di utenze su vari servizi esterni. Ogni utenza della piattaforma ha in particolare associato un record su:

- **Firebase Authentication**: per la parte vera e propria di registrazione, autenticazione e gestione delle sessioni
- **CloudSQL**: dove vengono memorizzati i dettagli principali relativi all'utenza
- **Mailchimp**: dove viene creata un'utenza utile ai fini di marketing e engagement
- **Stripe**: dove viene creata un'utenza a cui vengono collegati i vari pagamenti effettuati/ricevuti e i dettagli di fatturazione

Ad ogni utenza viene assegnato il ruolo a cui appartiene, per poi consentire l'accesso alla singola piattaforma dedicata, ad esempio un negozio può autenticarsi solamente sul pannello a lui dedicato, così come amministratori, utenti e comuni. Questa funzionalità è messa a disposizione da Firebase Authentication attraverso l'utilizzo dei *custom claims*, ovvero un oggetto che al suo interno contiene dati di tipo chiave-valore assegnati in fase di registrazione e utilizzati per distinguere i vari tipi di utenza.

Identificatore	Provider	Data di creazione	Accesso eseguito	UID utente ↑
ledremopsa@nedoz.com +393486528152		5 ott 2020	27 nov 2020	09aSB8dYNjfFdI5ZaNnepAfpriM2

Figura 4.1: Record utenza Garzone su Firebase Authentication

Dal pannelo di amministrazione è possibile abilitare/disabilitare l'accesso alle varie piattaforme per ogni utente registrato. Questa operazione avviene cambiando il valore del flag *«abilitato»* presente in ogni istanza di utenza sulla base di dati e controllando ad ogni cambio di route l'effettiva abilitazione. Ogni richiesta effettuata dal client alla componente backend viene recepita e accettata solo nel caso in cui la sorgente disponga delle autorizzazioni necessarie, altrimenti è compito della singola function restituire una risposta provvista di errore.

4.1.1 Registrazione

La procedura di registrazione prevede, per i pannelli di negozi e utenti, una fase di landing su una precisa pagina dell'applicativo accessibile da tutti i client che ne fanno richiesta. Dopo la compilazione dei campi e la loro validazione è compito della componente di backend eseguire le seguenti operazioni:

1. Destrutturare i parametri della richiesta
2. Ri-validare i parametri della richiesta
3. Interfacciarsi al servizio di Firebase Authentication per la creazione dell'utenza
4. Invio della mail di verifica dell'account
5. Interfacciarsi eventualmente al servizio di geolocalizzazione per il computo delle coordinate geografiche dato l'indirizzo in un parametro della richiesta
6. Generare l'istanza dell'utenza provvista dei dettagli forniti nei parametri della richiesta e inserirla nel database relazionale
7. Assegnare i custom claims all'utenza generata su Firebase
8. Interfacciarsi al servizio esterno Mailchimp per la creazione dell'utenza associata
9. Restituire alla componente frontend il risultato dell'operazione

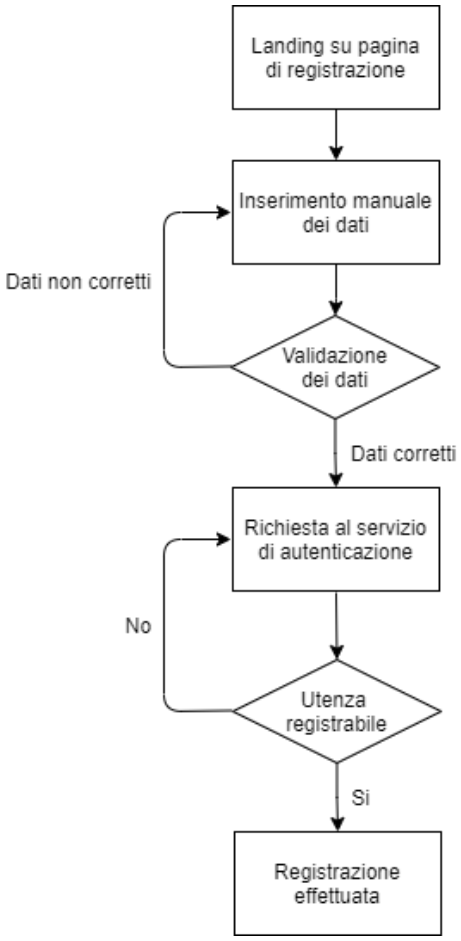


Figura 4.2: Flusso di registrazione di un negozio

La procedura di registrazione, per questioni di sicurezza non è prevista per comuni e amministratori. Questa operazione per l'amministratore superuser è stata prevista inizialmente, mentre per i comuni viene eseguita dal pannello di amministrazione.

4.1.2 Accesso

La fase di accesso, presente su ogni pannello, prevede l'inserimento delle credenziali di accesso fornite dall'utente in fase di registrazione. Il servizio backend fornito da Firebase Authentication provvede poi a verificarle e a fornire una risposta al client che ha richiesto l'accesso. Per ottenere l'accesso alla piattaforma sono necessari tre requisiti:

- Registrazione effettuata con successo
- Custom claim relativo al ruolo dell'utenza corrispondente al pannello dove si vuole effettuare la procedura di accesso
- Email già verificata in precedenza
- Account abilitato su firebase
- Account abilitato sulla base di dati

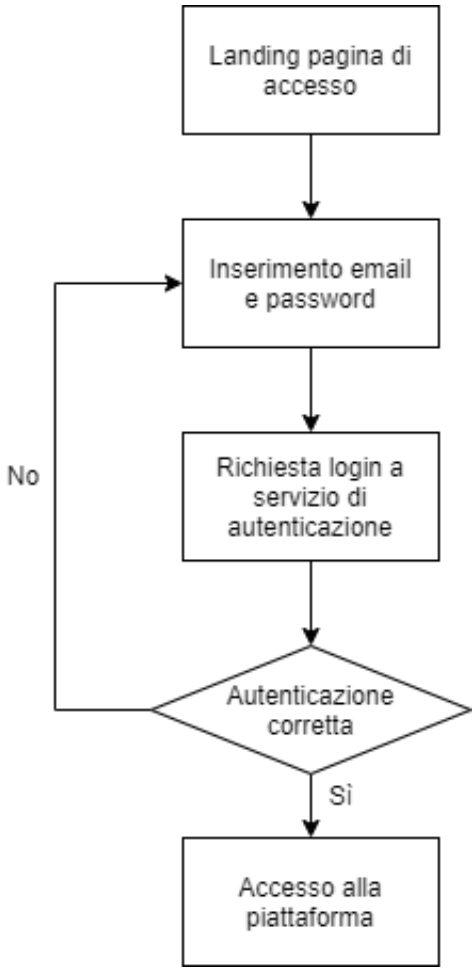


Figura 4.3: Flusso di login del negozio

Per una corretta gestione dei claims è presente, in tutti gli applicativi frontend, un listener (procedura che in runtime resta in ascolto di un endpoint del backend) che ha il compito di verificare a intervalli costanti i permessi dell'utenza e, nel caso non fossero coerenti, di effettuare la procedura di logout.

Private Route Nell'applicativo, ad ogni navigazione verso pagine diverse dalla corrente, è previsto, per le route private, una serie di controlli per evitare che un utente non autorizzato acceda a sezioni di Garzone a lui non consentite. Questa funzionalità è prevista dal componente `PrivateRoute`, che renderizza il componente `Route` (lo screen richiesto), solo dopo aver effettuato tutti i con trolli necessari.

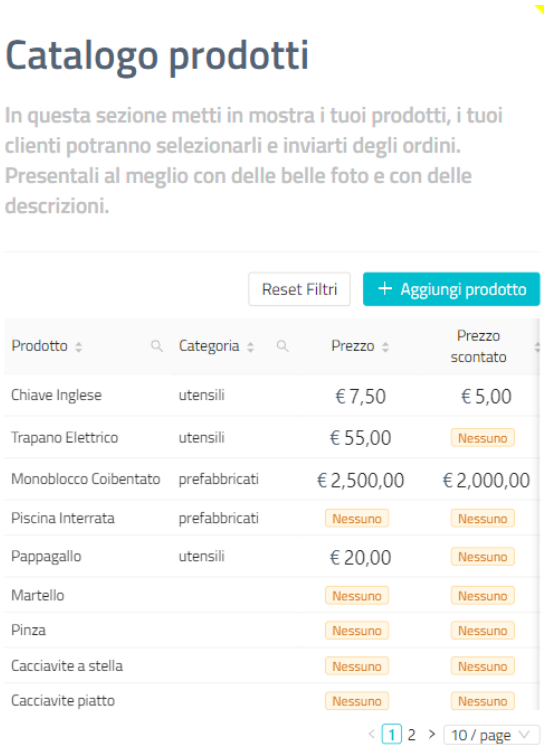
```
// import router
import { Route, Redirect } from "react-router-dom";

class PrivateRoute extends React.Component {
  is_mounted = false;

  constructor(props) {
    super(props);
    this.state = {
      not_found: false,
      loading: true,
    };
  }

  componentDidMount() {
    this.is_mounted = true;
  }

  componentWillUnmount() {
    this.is_mounted = false;
  }

  render() {
    return this.props.notVerified ? (
      <Redirect to={"/conferma-mail"} />
    ) : this.props.abilitato === false && this.props.authed === true ? (
      <Redirect to={"/non-abilitato"} />
    ) : this.props.funzioneNonAbilitata === true && this.props.authed === true ? (
      <Redirect to={"/funzione-non-abilitata"} />
    ) : this.props.authed == true ? (
      <Route {...this.props} />
    ) : this.props.location.pathname != "/conferma-mail" ? (
      <Redirect
        to={{ pathname: "/accedi", state: { from: this.props.location } }}
      />
    ) : (
      <Route {...this.props} />
    );
  }
}
```

Figura 4.4: Componente `PrivateRoute`

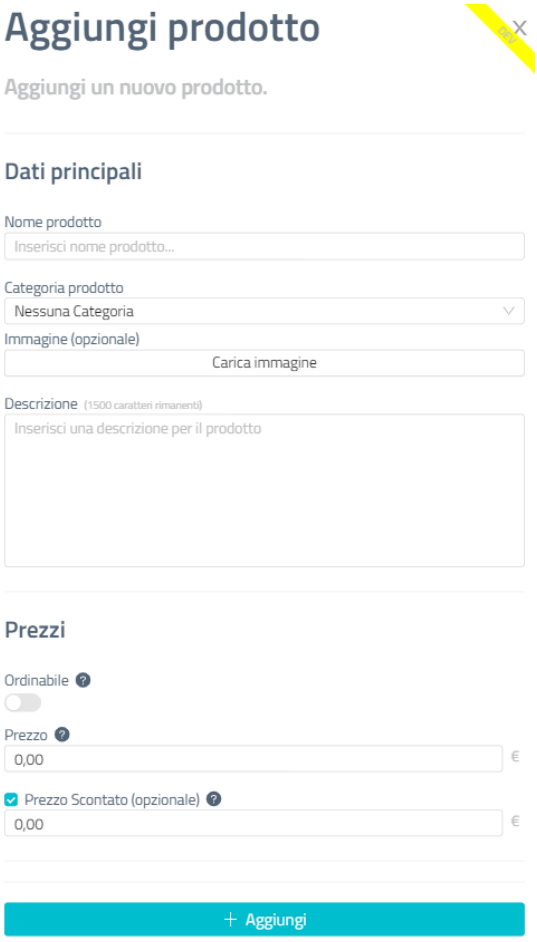
4.2 Catalogo Prodotti e Catalogo Servizi

Ogni negoziante ha la possibilità, tramite il proprio pannello, di inserire e gestire sia prodotti che servizi del suo catalogo. Successivamente i record generati possono essere visionati dagli utenti dall'apposita piattaforma e gestiti inoltre dall'amministrazione.

Entità Un prodotto o un servizio da registrare necessita di dati relativi a nome, descrizione, categoria, immagine, prezzo, prezzo scontato e flag di messa in vendita.



(a) Gestione prodotti - Pannello negozio



(b) Aggiunta prodotto - Pannello negozio

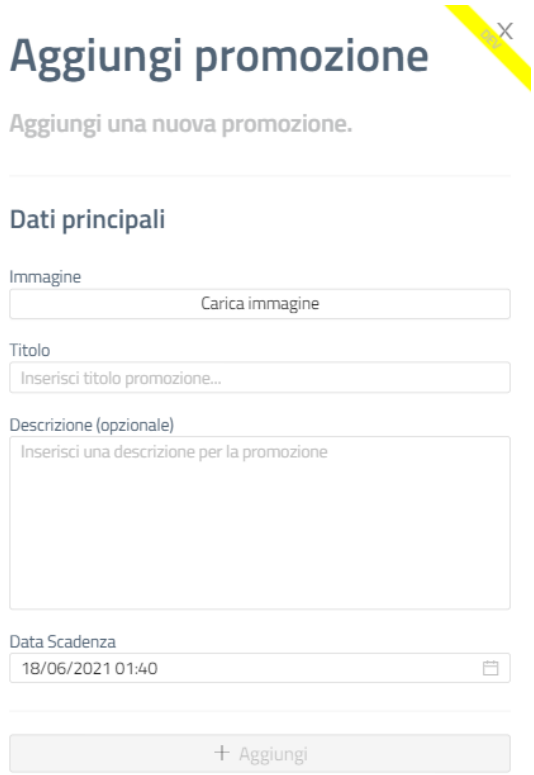
Figura 4.5: Schermate di Garzone relaitive al catalogo prodotti

Dal pannello di gestione è inoltre prevista la funzionalità di ricerca e filtraggio dei dati in base a parametri scelti dall'utente. Per la fase di registrazione di prodotti e servizi, data la necessità di allegare immagini, è previsto l'utilizzo del bucket di Firebase. In caso di cancellazione del record è necessaria quindi l'eliminazione del file associato all'immagine dal bucket stesso.

4.3 Promozioni

Altra funzionalità fornita ai negozi è relativa alla creazione e gestione di promozioni, ovvero annunci visionabili dagli utenti provvisti di data di scadenza. Per la gestione delle date è stata utilizzata la libreria `moment.js`, che fornisce semplificazioni riguardo le operazioni da eseguire sulla validazione dei dati in fase di creazione e di modifica.

Entità Una promozione necessita di dati relativi a titolo, descrizione, data di scadenza e un immagine esplicativa.



(a) Aggiunta promozione - Pannello negozio

(b) Promozione - Garzone

Figura 4.6: Schermate di Garzone relative alle promozioni

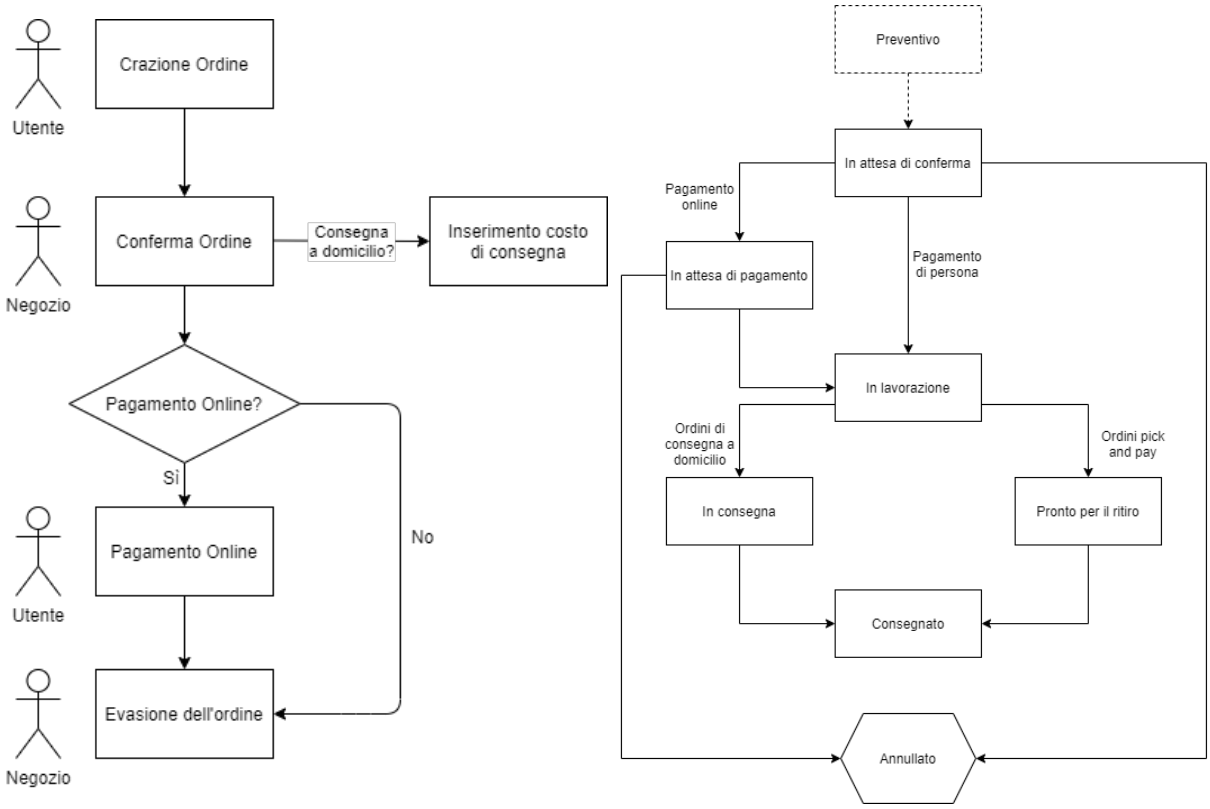
4.4 Ordini

Una delle funzionalità più rilevanti all'interno di Garzone è sicuramente quella relativa alla creazione e alla gestione degli ordini. Un ordine viene generato da un utente da un'iniziale composizone di un carrello con prodotti selezionati dalla navigazione tra i cataloghi di un negozio. Dopo aver composto il carrello è possibile per l'utente procedere con la creazione dell'ordine, inserendo dati relativi a modalità di consegna, modalità

di pagamento, eventuali note per il negozio ed eventuale indirizzo di consegna. Sarà poi compito del negozio confermare l'ordine ricevuto e procedere con la sua evasione, richiedendo eventualmente il pagamento online per gli ordini che lo prevedono. Il negozio

Figura 4.7: Form di creazione ordine

ha il compito infine di gestire l'avanzamento di stato dell'ordine dal proprio pannello di controllo, terminandone l'evasione selezionando il relativo status "consegnato". Ad ogni passaggio di status dell'ordine, la componente di backend ha il compito di notificare correttamente sia l'utente sia il negozio coinvolti tramite notifiche push e email transazionali.



(a) Flusso di gestione di un ordine

(b) Diagramma dei possibili stati di un ordine

Figura 4.8: Schermate di Garzone relative alle promozioni

4.4.1 Preventivi

Un ordine può essere anche generato da un negozio. Per eseguire questa operazione è necessario che il cliente richieda un preventivo al negozio. L'implementazione di questa funzionalità è stata prevista richiedendo ad un utente di inserire i dati necessari alla sua creazione, per poi generare un messaggio di chat che, invece di essere di tipo *text*, è di tipo *preventivo*. Il componente che effettua il render del messaggio di chat lato negozio interpreta automaticamente il messaggio di tipo preventivo e offre al negozio la possibilità di generare un preventivo tramite un modulo precompilato con i dati inseriti dall'utente. L'ordine sarà creato nello status *preventivo* ed è questa volta compito dell'utente confermare o rifiutare il preventivo. In caso di divergenze tra domanda e offerta è possibile per il negozio modificare il preventivo creato e comunicarlo all'utente per poi consentire il preseguimento dell'evasione dell'ordine.

4.5 Chat

La Chat di Garzone permette di far comunicare tra loro un negozio ed un cliente (One to One chat). L'idea alla base è quella di sfruttare in combinazione Firebase Realtime Database e Cloud SQL al fine di raggiungere il giusto compromesso tra performance, tempi di sviluppo e gestione dei costi. Firebase realtime Database si occuperà di salvare per ogni utente il counter delle notifiche e la sua lista di chat, mentre Cloud SQL salverà i messaggi (opzionalmente anche i contenitori chat). Firebase Realtime Database salverà per ogni utente, la lista delle sue chat oltre ai contatori di notifiche, in questo modo potrà essere sfruttata la funzionalità "real-time" per tenere aggiornata la lista chat e i badge di notifiche su tutti i client.
Secondo quanto sopracitato avremo un database che seguirà il qui descritto schema:

```
{
  "users": {
    // user 1 (è un utente) Mario Rossi
    "firebase_user_id_1": {
      "chats_notification_counter": 0,
      "chats": {
        "chat_1_autogenerate_realtime_database_id": {
          "chat_sql_id": "chat_sql_id",
          "to_read" : true,
          "last_update": "TIMESTAMP",
          "last_message": "Text for the last message",
          "chat_title": "Negozio 1",
        }
      }
    },
    // user 2 (è un negozio) Negozio 1
    "firebase_user_id_2": {
      "chats_notification_counter": 0,
      "chats": {
        "chat_1_autogenerate_realtime_database_id": {
          "chat_sql_id": "chat_sql_id",
          "to_read" : false,
          "last_update": "TIMESTAMP",
          "last_message": "Text for the last message",
          "chat_title": "Mario Rossi",
        }
      }
    },
  }
}
```

Figura 4.9: Record di chat in Firebase Realtime Database

4.6 Appuntamenti

Per la gestione appuntamenti è stato previsto l'utilizzo del package `react-big-calendar` che offre un'interfaccia grafica ricca di funzionalità per creare e modificare eventi. Un utente può far uso dell'apposita funzionalità per richiedere un appuntamento ad un negozio. Viene quindi generato un messaggio di chat provvisto di richiesta, il quale

viene automaticamente inviato al negozio. Sarà compito di quest'ultimo può creare l'appuntamento collegato al cliente che ne ha fatto richiesta. Nella pagina del negozio lato utente è possibile consultare gli slot orari disponibili, ovvero ancora non occupati dalla generazione di altri eventi, infatti in fase di creazione è previsto un filtraggio dei soli slot disponibili alla prenotazione.

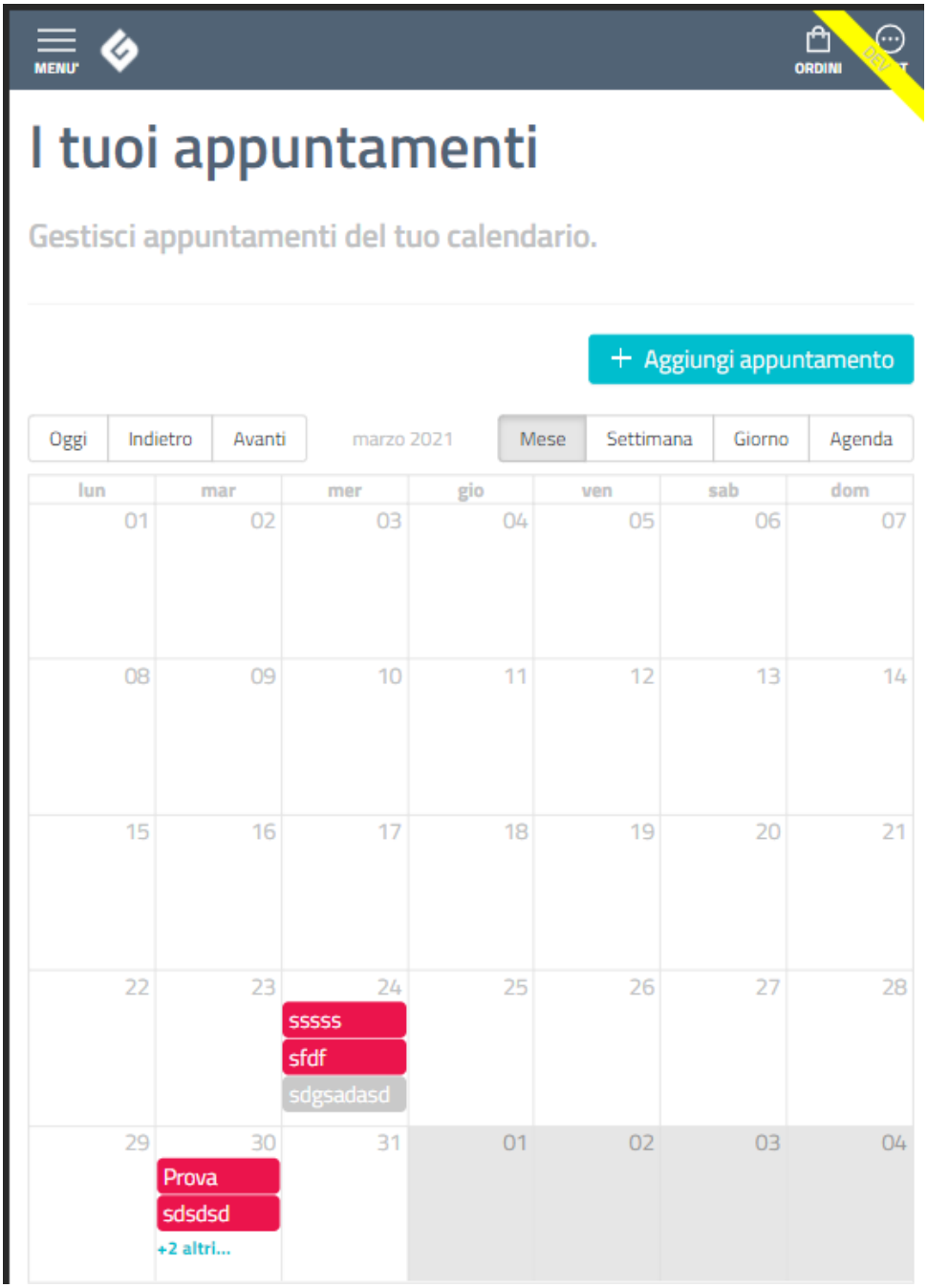


Figura 4.10: Panoramica appuntamenti - Pannello negozio

Capitolo 5

Sicurezza degli applicativi

5.1 Integrità del dato

5.1.1 Yup: Validazione frontend e backend

Per ottenere uno standard qualitativo sui dati inseriti nel database è prevista una stringente e doppia validazione. Infatti sia lato backend che lato frontend è presente una serie di controllo che impediscono inserimenti di dati non conformi alle aspettative. Questa funzionalità è offerta dal package `Yup`, uno schema builder per il parsing e la validazione di dati.

5.1.2 Transazioni SQL

Nella componente backend è tra le altre cose prevista l'implementazione di transazioni SQL. Una transazione offre la possibilità di creare un punto di ripristino nel caso in cui delle operazioni SQL concatenate non vadano a buon fine. Le transazioni in Garzone sono utilizzate soprattutto in funzioni concernenti pagamenti ed eliminazioni/modifiche/aggiunte a cascata di record, come ad esempio la creazione di un utenza Mailchimp e stripe collegata ad un account di cliente.

```
try {
  await pool.query("START TRANSACTION");
  await pool.query(
    `DELETE FROM integrazioni_stripe WHERE id_negozio = ?`,
    id
  );
  await pool.query(`INSERT INTO integrazioni_stripe SET ?`, [
    integrazioneRecord,
  ]);
  await pool.query("COMMIT");
} catch (error) {
  console.log(error);
}
```

Figura 5.1: Implementazione in Garzone di una transazione

ACID

L'implementazione delle transazioni gode di un insieme di proprietà dette ACID (Atomicity, Consistency, Isolation, Durability)[13].

- **Atomicità**, la quale prevede che una transazione sia un'unità indivisibile di operazione
- **Consistenza**, la quale prevede che l'esecuzione della transazione non violi i vincoli di integrità definiti sul database
- **Isolamento**, che prevede che l'esecuzione di una transazione sia indipendente dall'esecuzione contemporanea di altre transazioni
- **Persistenza**, che prevede che l'effetto di una transazione che ha eseguito il commit correttamente non venga più perso

5.2 Sicurezza online

In Garzone è prevista anche una componente relativa alla sicurezza online, di vitale importanza per evitare che malintenzionati manomettano il corretto funzionamento di tutte le componenti parte dell'applicativo.

5.2.1 Whitelist chiamate API

Una prima protezione riguarda l'interfacciamento dei vari servizi terzi ai vari pannelli. Per ottenere un adeguato livello di protezione è stato previsto che le chiamate verso i servizi Google siano consentite solo dai domini su cui sono presenti i vari pannelli. Un

malintenzionato, nel caso in cui dovesse venire in possesso della chiave API che consente le chiamate ai servizi Google, dovrebbe effettuare l'ulteriore operazione di sostituirsi all'host sorgente abilitato alle chiamate.

5.2.2 Firebase Functions Context

Firebase Authentication mette a disposizione lato backend tra i parametri di una function quello relativo al `context`, che permette di ottenere e analizzare diversi attributi riguardo la chiamata alla Cloud Function. In Garzone questa funzionalità è servita per validare la corrispondenza tra l'`uid` (Unique Identifier dell'utente) fornito nei parametri della richiesta e l'`uid` fornito dal `context`. In caso i due valori non dovessero coincidere il sistema backend restituisce come risultato un'apposita eccezione `not-authorized`.

5.2.3 Webhook

Un'altra tecnica per fornire un adeguato livello di sicurezza agli applicativi prevede l'utilizzo di WebHook da parte di servizi esterni per effettuare operazioni sensibili, come il processo di pagamento. Infatti per processare i pagamenti in Garzone è delegata a Stripe la chiamata alla Cloud Function che si occupa di cambiare stato all'ordine. In questo modo è impossibile per un utente malintenzionato effettuare operazioni di cambio stato dell'ordine fingendo un pagamento non effettuato.

5.2.4 Security Rules

Per quanto riguarda la sicurezza del servizio di Firebase Realtime Database, è stato previsto l'inserimento di regole di sicurezza basati su ciò che un'utenza può ottenere/-modificare/eliminare. In Garzone è imposto che ad ogni utenza è possibile accedere solo all'oggetto di Realtime Database che presenta come chiave il proprio `uid`, ovvero il suo identificativo. Così è impossibile per un'utenza effettuare operazioni malevole su dati relativi ad altre utenze. Le security rules sono state previste anche per il servizio di Cloud Storage prevedendo le stesse caratteristiche imposte sulle utenze riguardo le regole di Realtime Database.

Capitolo 6

Conclusioni

6.1 Dati sull'utilizzo

Prima del suo rilascio, è stata offerta l'opportunità al comune di Muggiò (MB) e al comune di Santa Croce Camerina (RG) di poter usufruire delle prime funzionalità, iniziando con la semplice registrazione dei negozi e l'inserimento di prodotti e servizi. Nei successivi rilasci sono state implementate le funzionalità restanti, di cui usufruiscono, alla data di Giugno 2021, 11 comuni attualmente registrati.

- Città di Meda
- Comune di Busto Garolfo
- Comune di Cantù
- Comune di Cologno Monzese
- Comune di Giussano
- Comune di Muggiò
- Comune di Pianezza
- Comune di Ponte san Pietro
- Comune di Santa Croce Camerina
- Comune di Sondrio
- Distretto Commerciale di Pioltello

Inoltre, il bacino di utenza dei negozianti conta, alla data di Giugno 2021, 779 negozi iscritti.

6.2 Sviluppi futuri

Gli sviluppi futuri della piattaforma includono, tra le altre cose, il rilascio di un'app mobile lato cliente e un'app mobile lato negoziante, riprendendo in toto le funzionalità implementate nella soluzione web. Alla data di Giugno 2021 è stata già rilasciata l'anteprima dell'applicazione mobile riscontrando grande successo nei vari store di destinazione.

Bibliografia

[1] Ministero per l'innovazione tecnologica e la transizione digitale. *Solidarietà Digitale al servizio di studenti e commercianti.* https://solidarietadigitale.agid.gov.it/iniziative/, 2020.

[2] UNIONCAMERE. *Covid: 7 idee innovative per la ripartenza Premiati a Maker Faire i vincitori di Top of the Pid.* https://www.puntoimpresadigitale.camcom.it/paginainterna/premio-top-of-the-pid, 2020.

[3] Team per la trasformazione digitale, Ministero per l'innovazione tecnologica e la transizione digitale. *Linee guida di design per i servizi web della PA.* https://docs.italia.it/italia/design/lg-design-servizi-web/it/bozza/index.html, 2020.

[4] Jacobson I., Booch G., Rumbaugh J. *The Unified Software Development Process.* Addison-Wesley, 1999.

[5] Rational Software. *Rational Software White Paper.* IBM, 18880 Homestead Road, Cupertino, CA 95014, tp026b, rev 11/01 edition, 1998.

[6] Marijn Haverbeke. *Eloquent Javascript.* Editore unico Hoepli Milano, via Hoepli 5, 20121 Milano, 2 edition, 2015.

[7] airbnb. *Airbnb JavaScript Style Guide().* https://github.com/airbnb/javascript, 2012.

[8] Wojciech Maj. *react-lifecycle-methods-diagram.* https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/, 2018.

[9] Ant Design Team. *ant-design.* https://ant.design/docs/react/introduce, 2015.

[10] Google. *Firebase.* https://firebase.google.com/products-release, 2021.

[11] Amazon AWS. *Simple Email Service.* https://aws.amazon.com/it/ses/, 2021.

[12] Stripe. *Stripe Connect*. `https://stripe.com/it/connect`, 2021.

[13] Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone. *Basi di Dati, Modelli e linguaggi di interrogazione*. The McGraw-Hill Companies, srl, via Ripamonti, 89 - 20139 Milano, terza edizione edition, 2009.