

Documentazione NebulaWatches

Titolo del progetto: NebulaWatches
Alunni: Alexandru Ciobanu, Gioele Chiodoni, Tom Schillerwein
Classe: Info 3BC, 3BB
Anno scolastico: 2023/2024
Docente responsabile: Guido Montalbetti

| | | |
|-------|--|----|
| 1 | Introduzione | 3 |
| 1.1 | Informazioni sul progetto | 3 |
| 1.2 | Abstract | 3 |
| 1.3 | Scopo | 3 |
| 2 | Analisi | 4 |
| 2.1 | Analisi del dominio | 4 |
| 2.2 | Analisi e specifica dei requisiti | 4 |
| 2.3 | Use case | 7 |
| 2.4 | Pianificazione | 8 |
| 2.5 | Analisi dei mezzi | 8 |
| 2.5.1 | Software | 8 |
| 2.5.2 | Hardware | 8 |
| 3 | Progettazione | 9 |
| 3.1 | Design dell'architettura del sistema | 9 |
| 3.2 | Design dei dati e database | 11 |
| 3.3 | Design delle interfacce | 11 |
| 3.4 | Design procedurale | 12 |
| 4 | Implementazione | 14 |
| 4.1.1 | MySQL | 14 |
| 4.1.2 | Node | 14 |
| 4.1.3 | Java | 14 |
| 4.1.4 | Maven | 14 |
| 4.1.5 | Python3 | 15 |
| 4.1.6 | Servizi | 15 |
| 4.2 | Frontend | 17 |
| 4.2.1 | Struttura View | 17 |
| 4.2.2 | Codice | 18 |
| 4.2.3 | Interfaccia | 20 |
| 4.3 | Backend | 20 |
| 4.3.1 | Security | 21 |
| 4.3.2 | Storage | 23 |
| 4.3.3 | Team | 24 |
| 4.3.4 | Client | 24 |
| 4.3.5 | Watches | 24 |
| 4.3.6 | Utils | 25 |
| 4.3.7 | Chatbot | 26 |
| 5 | Test | 31 |
| 5.1 | Protocollo di test | 31 |
| 5.2 | Risultati test | 43 |
| 5.3 | Mancanze/limitazioni conosciute | 61 |
| 6 | Consuntivo | 61 |
| 7 | Conclusioni | 62 |
| 7.1 | Sviluppi futuri | 62 |
| 7.2 | Considerazioni personali | 62 |
| 8 | Glossario | 63 |
| 9 | Bibliografia | 64 |
| 9.1 | Sitografia | 64 |
| 10 | Allegati | 64 |
| 11 | Indice Delle Figure | 64 |

1 Introduzione

1.1 Informazioni sul progetto

- Titolo Progetto: NebulaWatches
- Allievi: Alexandru Ciobanu I3BB, Gioele Chiodoni I3BB, Tom Schillerwein I3BC
- Docente responsabile: Guido Montalbetti
- Scuola: Arti e Mestieri Trevano, sezione Informatica
- Data di inizio e fine: 12.01.2024 – 03.05.2024
- Data di presentazione: 17 – 24 .05.2024

1.2 Abstract

For this project, a web application needs to be developed to manage a watch store.

Users have several basic management tools at their disposal, such as a list of watches from which they can compose their inventory, do advanced watch searches, and view their customers and what they buy.

They also have at their disposal advanced tools such as a chatbot, based on data from our Database, and a company message centralization point to improve and optimize the management of their watch store.

In addition, this project is about learning how to manage a project in the agile mode and how to work and share work in a team.

1.3 Scopo

Lo scopo di questo progetto è sviluppare un'applicazione web per la gestione di un negozio di orologi. Gli utenti avranno accesso a diversi strumenti di base, tra cui una lista di orologi per la composizione dell'inventario, ricerche avanzate sugli orologi e la visualizzazione dei clienti e dei loro acquisti.

Inoltre, saranno disponibili strumenti avanzati come un chatbot basato sui dati del database e un punto centrale per la gestione dei messaggi aziendali, al fine di ottimizzare l'efficienza operativa del negozio di orologi.

Infine, questo progetto serve anche a imparare a gestire un progetto nella modalità agile e a lavorare e spartirsi il lavoro in un team.

2 Analisi

2.1 Analisi del dominio

Abbiamo scelto di sviluppare un applicativo web che permetta all'utente di migliorare e semplificare la gestione del proprio negozio di orologi. Il prodotto deve essere facile da usare e presentare un'interfaccia grafica moderna. A nostro sapere non esiste un prodotto specifico per la gestione dei negozi di orologi sul mercato.

2.2 Analisi e specifica dei requisiti

| ID: REQ-001 | |
|-----------------|--|
| Nome | Accesso e login |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Un utente per poter utilizzare il sito deve effettuare il login. |
| Sotto requisiti | |
| 001 | Maschera di login |

| ID: REQ-002 | |
|-----------------|--|
| Nome | Registrazione |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Un utente senza account deve poter registrarsi tramite email oppure tramite Google |
| Sotto requisiti | |
| 001 | Maschera di registrazione |
| 002 | Se con email bisogna fare la verifica dell'email |

| ID: REQ-003 | |
|-----------------|---|
| Nome | Database |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Database contenente orologi, utenti, inventario dell'utente, clienti dell'utente e team dell'utente |

| ID: REQ-004 | |
|-----------------|--|
| Nome | Chatbot |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Chatbot basato su dati del database che aiuta e dà consigli all'utente |

| ID: REQ-005 | |
|-----------------|---------------------------------|
| Nome | Comunication center |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Centro di raggruppamento e-mail |

| ID: REQ-006 | |
|-----------------|--|
| Nome | Orologi nel magazzino |
| Priorità | 1 |
| Versione | 1.0 |
| Note | L'utente può selezionare gli orologi che ha nel proprio inventario/magazzino |
| Sotto requisiti | |
| 001 | Orologi venduti e comprati |
| 002 | Informazioni su spedizione, consegna, ecc. |

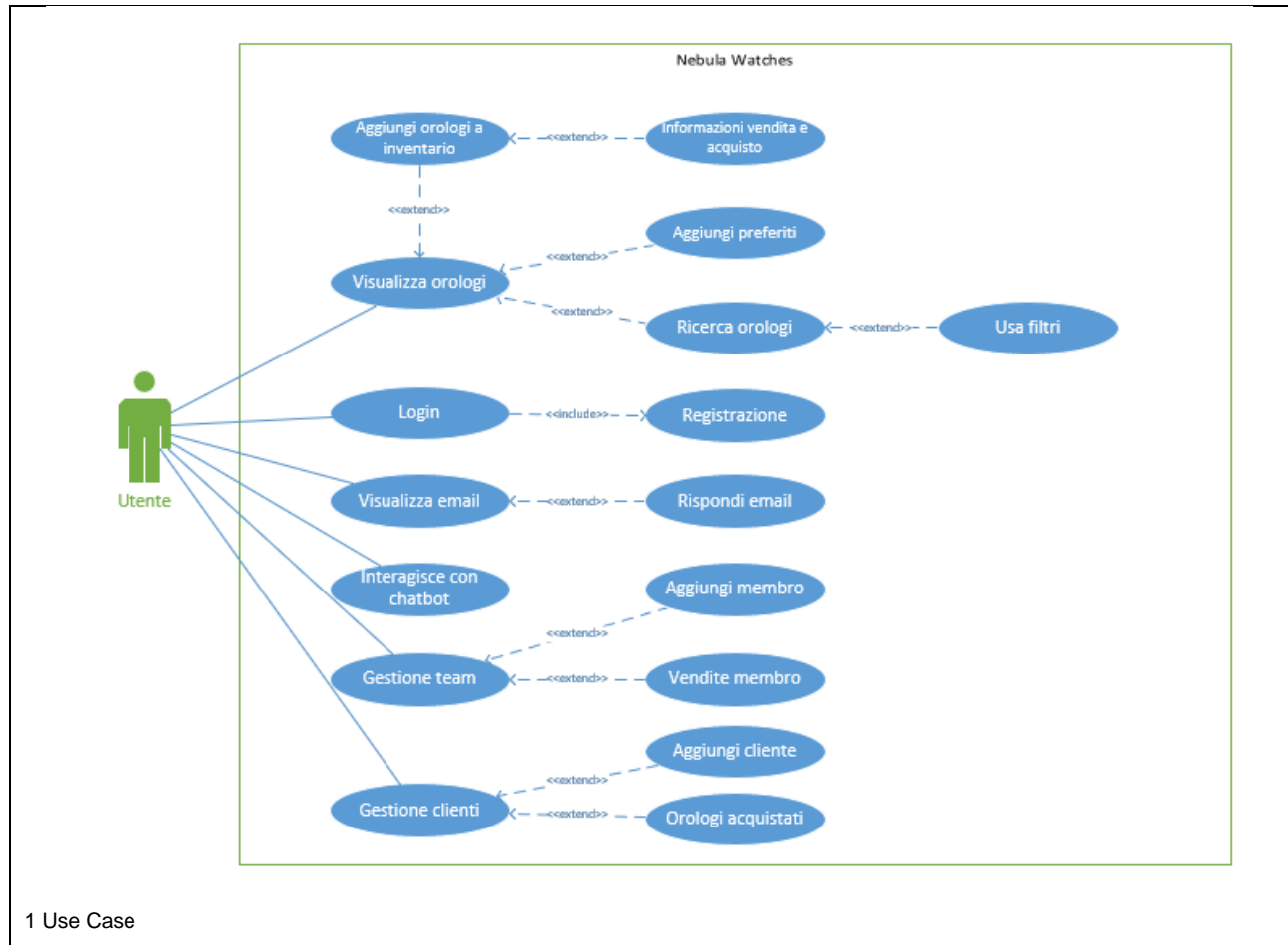
| ID: REQ-007 | |
|-----------------|--------------------------------------|
| Nome | Orologi |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Gli orologi sono ordinati per brand. |
| Sotto requisiti | |
| 001 | Ricerca e filtri di ricerca |
| 002 | Preferiti |

| ID: REQ-008 | |
|-----------------|--|
| Nome | Team |
| Priorità | 1 |
| Versione | 1.0 |
| Note | L'utente può aggiungere e gestire i propri membri del team |
| Sotto requisiti | |
| 001 | Informazioni di vendita dei membri del team |

| ID: REQ-009 | |
|-----------------|---------------------------------|
| Nome | Clienti |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Informazioni sui propri clienti |
| Sotto requisiti | |
| 001 | Orologi comprati |

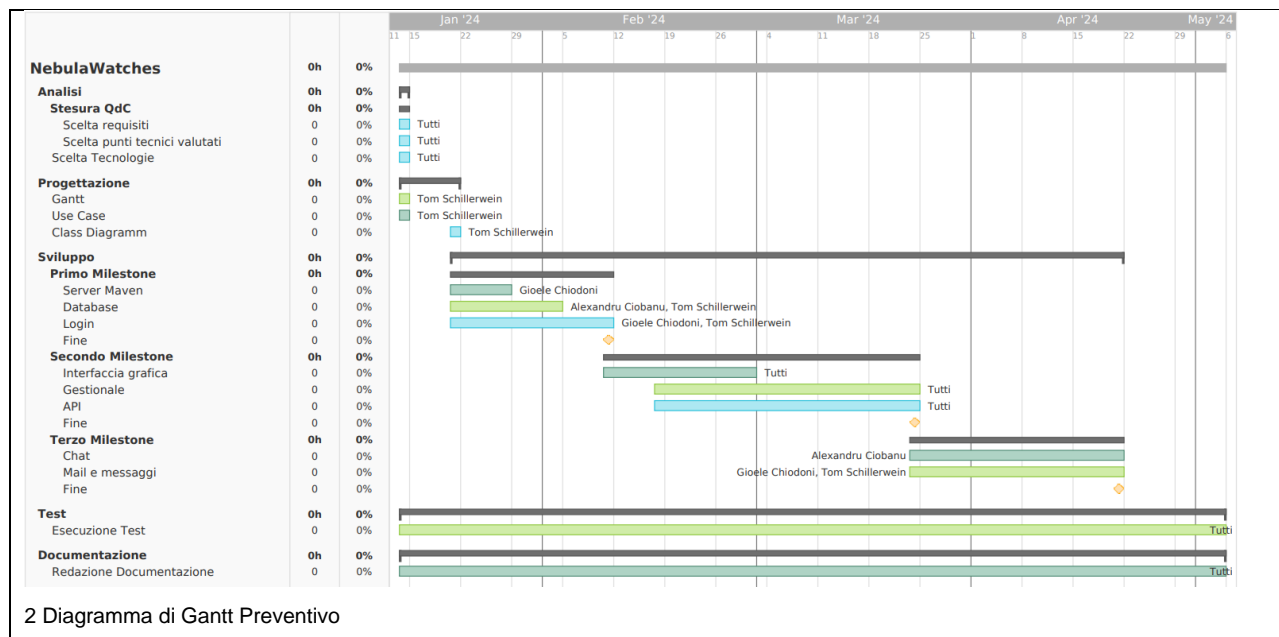
| ID: REQ-010 | |
|-----------------|-------------------------------------|
| Nome | Admin |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Gestione CRUD degli utenti |
| Sotto requisiti | |
| 001 | Paginazione e ricerca degli utenti. |

2.3 Use case



Come si può notare da questo schema, l'utente può eseguire una moltitudine di operazioni sull'applicativo. Tra questi si possono notare un login e la rispettiva registrazione, la gestione dei clienti e i suoi acquisti rispettivamente il proprio team e le loro vendite. Inoltre, può visualizzare una lista di orologi e comporre il proprio inventario, con le varie informazioni di acquisto e vendita. Infine l'utente può aggiungere gli orologi ai preferiti e eseguire delle ricerche su di essi.

2.4 Pianificazione



2.5 Analisi dei mezzi

Per questo progetto usiamo tre PC scolastici su cui eseguiamo delle macchine virtuali Linux Mint per lo sviluppo.

2.5.1 Software

- Visual Studio Code 1.85.2, come editor di testo
- IntelliJ IDEA Community Edition 2023.3.2, come IDE
- Google Chrome 116.0, come browser per visualizzare il sito durante lo sviluppo
- Mozilla Firefox 110.0, per testare la visualizzazione del sito
- Microsoft Project, per il Gantt
- Microsoft Visio, per fare lo Use Case
- Microsoft Word e Obsidian, per redigere la documentazione, il QdC e il diario
- GitHub, come Repository online
- Postman v10.22 per testare gli endpoint dell'API
- Trello.com, per la gestione dei task negli sprint

2.5.2 Hardware

Computer utilizzato come host delle macchine virtuali:

- Processore: Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz
- RAM: 32GB
- Scheda Video: NVIDIA GeForce RTX 2060
- SSD: 512GB
- 2 Display 1920x1080 60Hz

Virtual Machine su cui viene hostato il progetto:

- RAM: 4GB
- IP: 10.20.4.113

3 Progettazione

3.1 Design dell'architettura del sistema

Abbiamo progettato il nostro sistema per poter offrire un'esperienza completa agli utenti per la gestione del loro negozio di orologi.

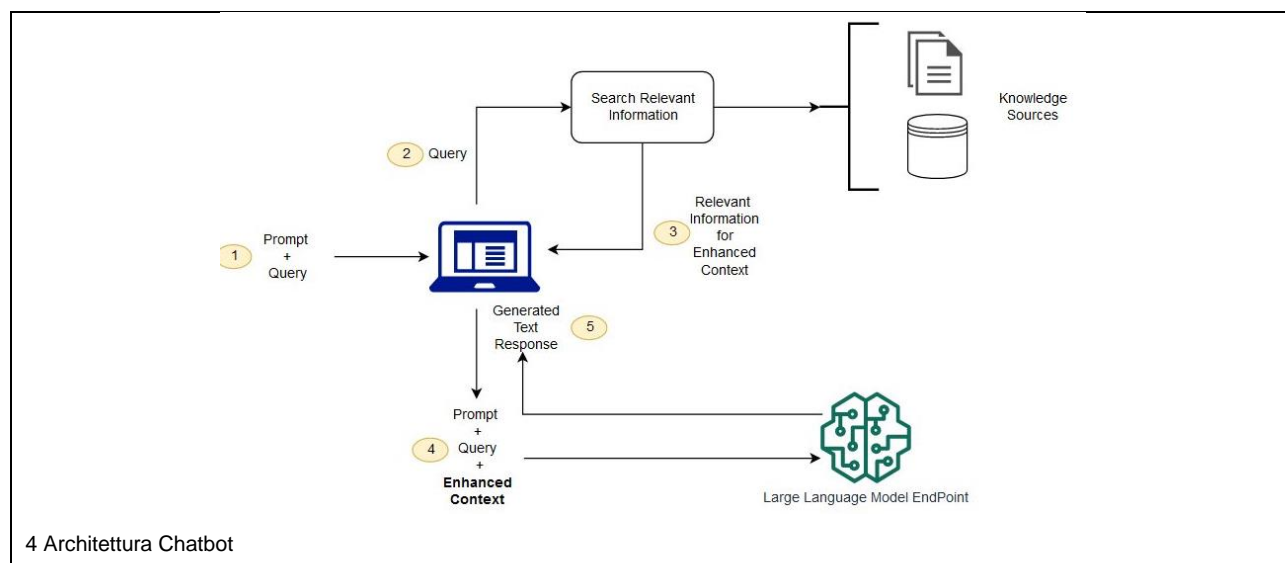
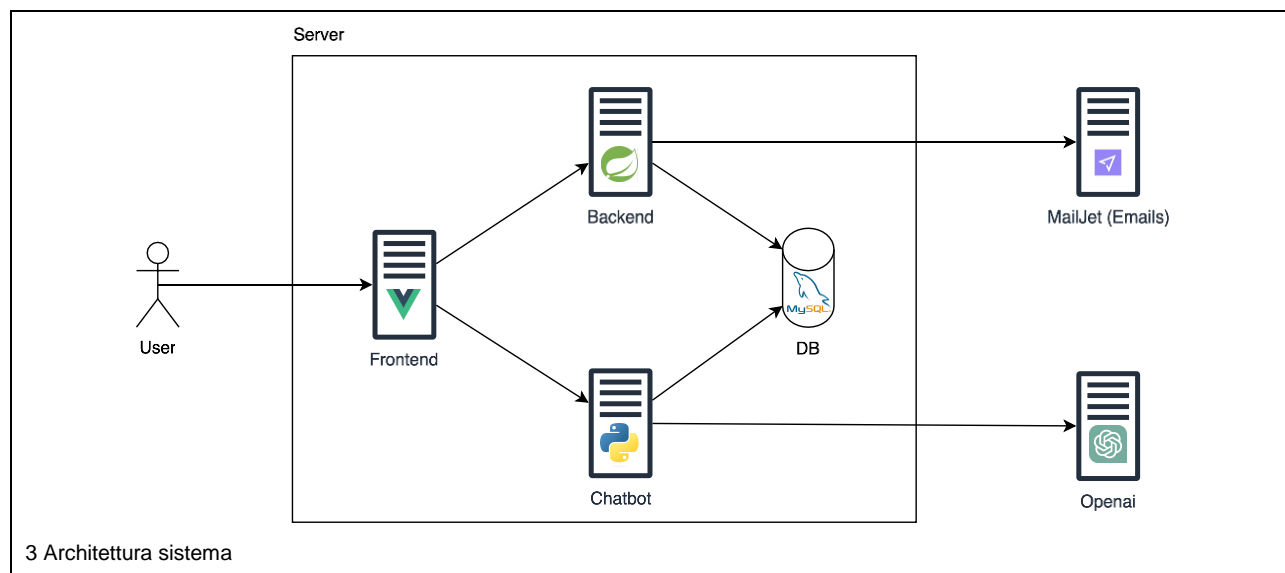
Gli utenti accedono all'applicazione tramite un'interfaccia grafica intuitiva e moderna, il frontend, che serve come punto di ingresso all'applicativo.

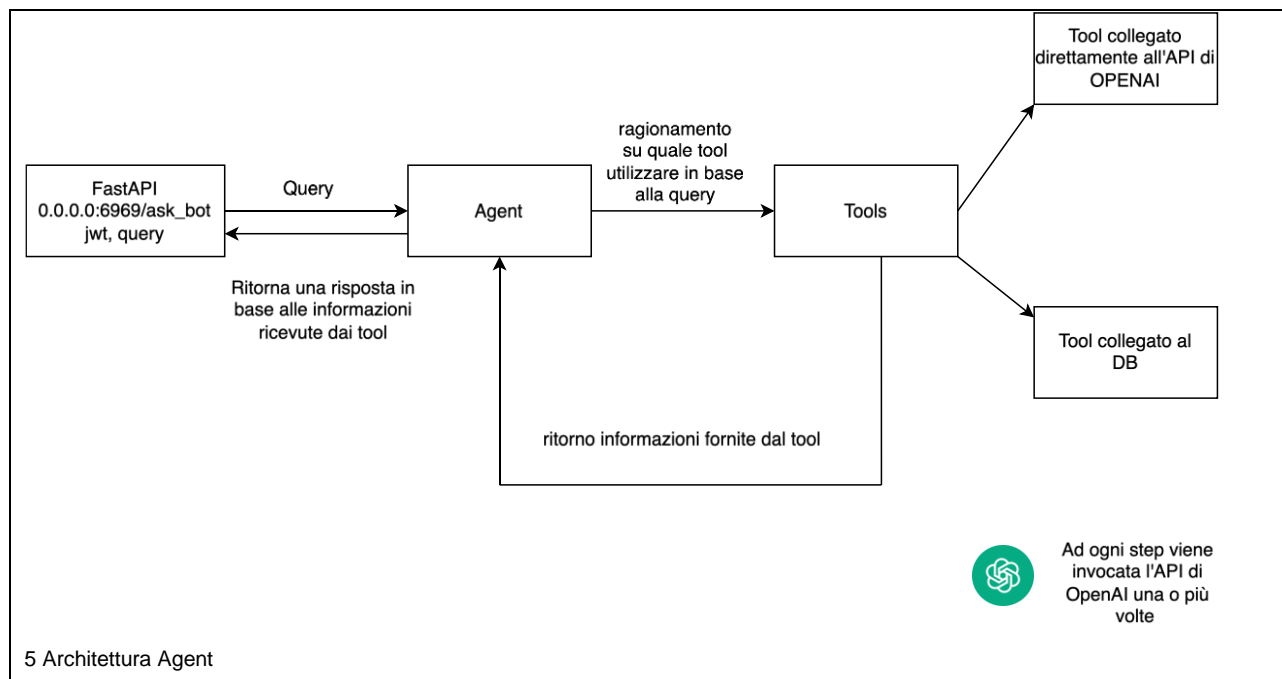
Le richieste degli utenti possono essere elaborate in due modi diversi: tramite il backend sviluppato con Java Spring o tramite un chatbot implementato con Python.

Java Spring gestisce molte funzionalità, incluse quelle che necessitano di un accesso al database MySQL per la gestione dei dati relativi agli orologi, ai clienti, agli acquisti e molto altro ancora.

Inoltre, il backend utilizza l'API di Mailjet per l'invio di e-mail agli utenti.

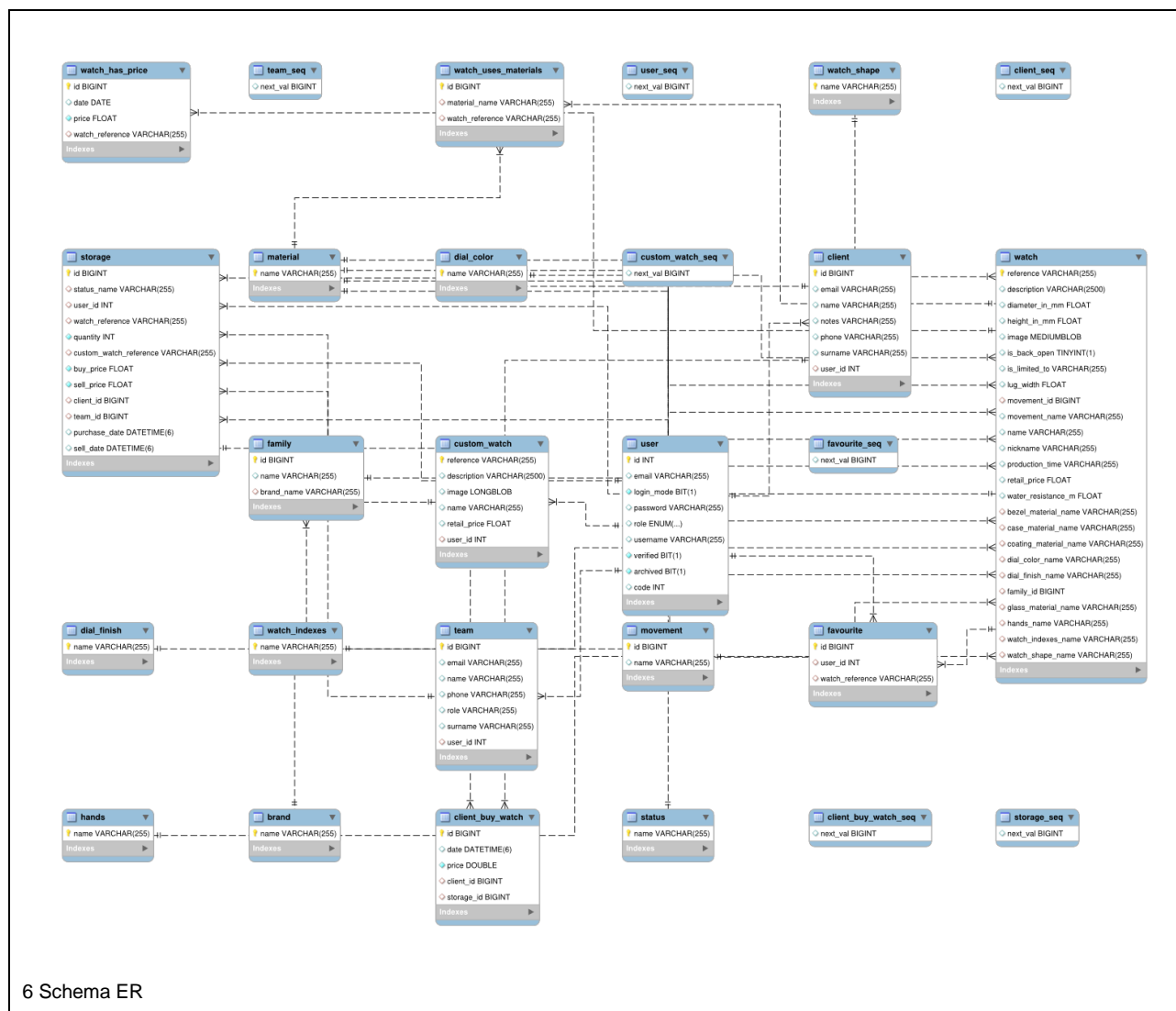
Il chatbot invece, offre un approccio interattivo e immediato per ricevere informazioni riguardanti la gestione del proprio negozio. Utilizzando l'API di OpenAI e le informazioni salvate nel database MySQL, il chatbot può fornire supporto personalizzato agli utenti, rispondendo a domande, fornendo raccomandazioni su orologi e assistendo nella gestione.





3.2 Design dei dati e database

Il nostro database è abbastanza complesso, ma la maggior parte delle tabelle servono per gli orologi. Non abbiamo definito noi in SQL il database, ma esso viene generato automaticamente da JPA, tramite Java Spring. Infatti, ci sono alcune tabelle di sistema come quelle che finiscono in “seq”. Le altre sono generate a partire dai Models presenti in Java, con i loro rispettivi collegamenti.



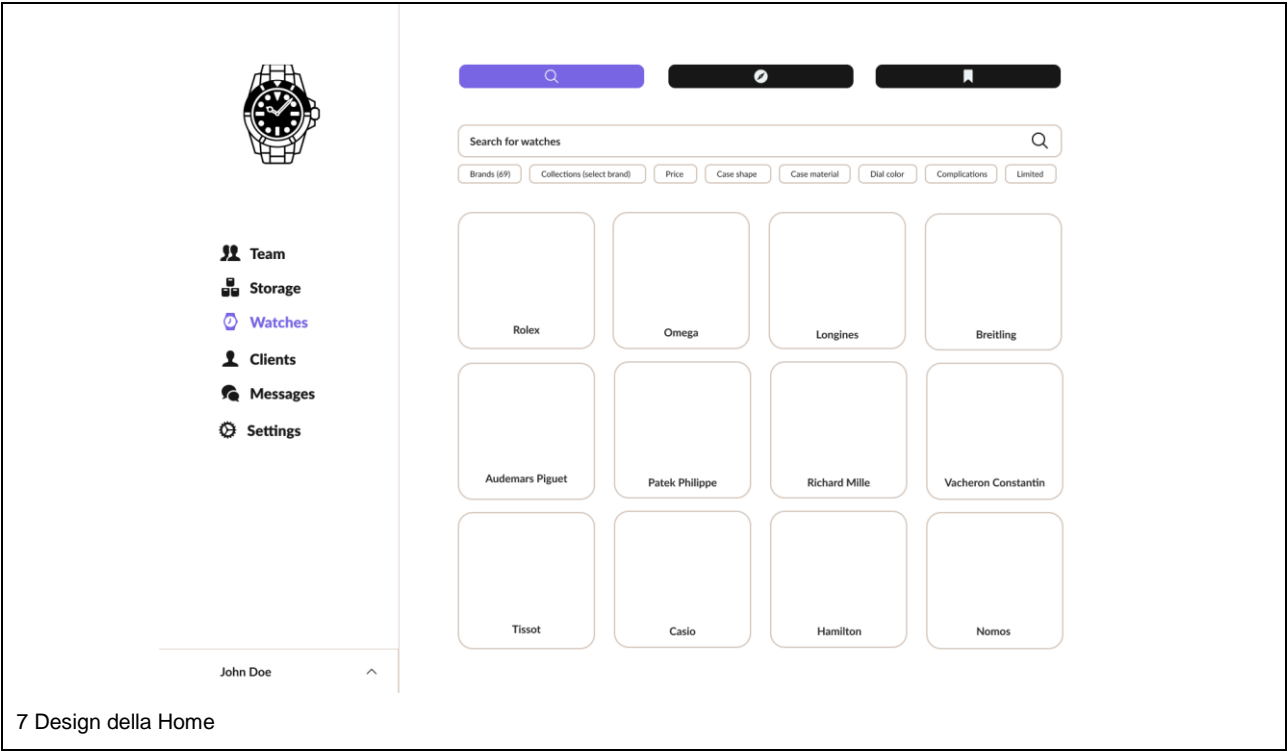
6 Schema ER

3.3 Design delle interfacce

Il design delle interfacce è un processo molto importante nella creazione di un'applicazione web, dato che qui si definiscono i colori, lo style e l'usabilità del sito web.

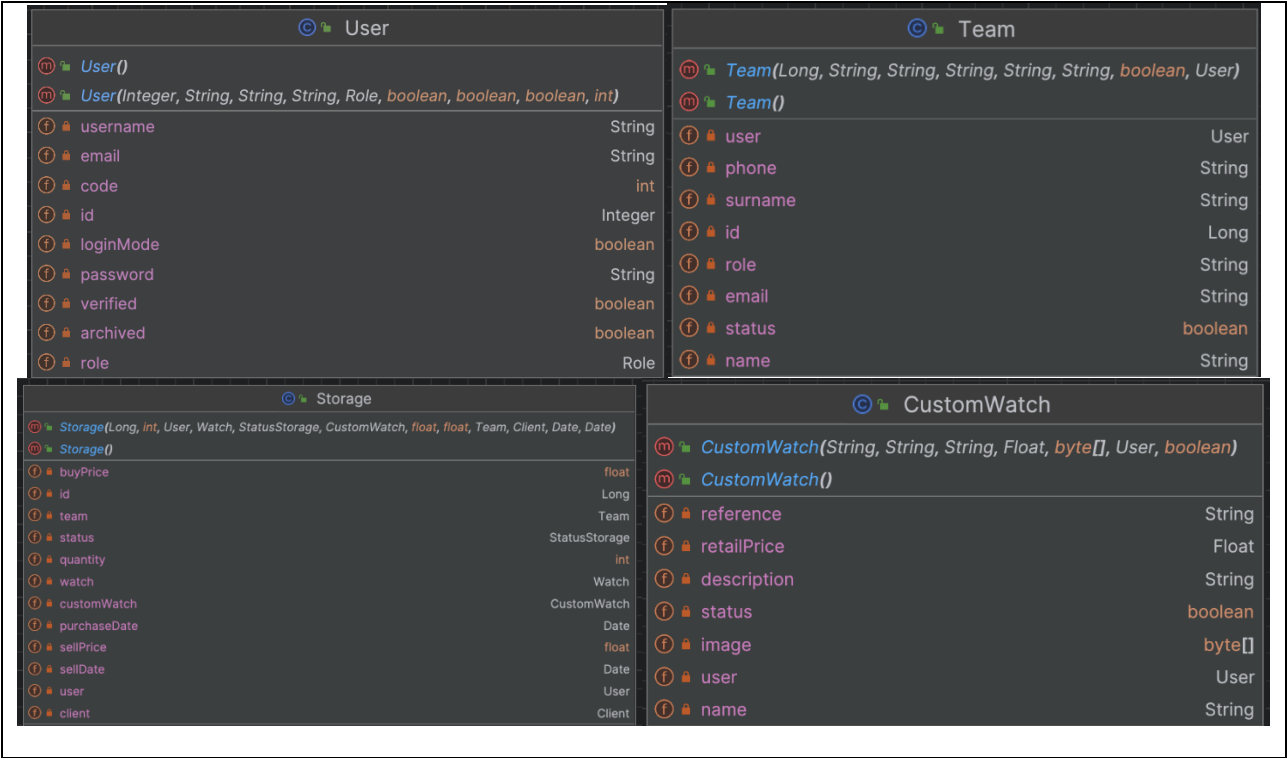
È stato realizzato solo il design della pagina Home, dato che le altre pagine saranno abbastanza simili nel design a questa.

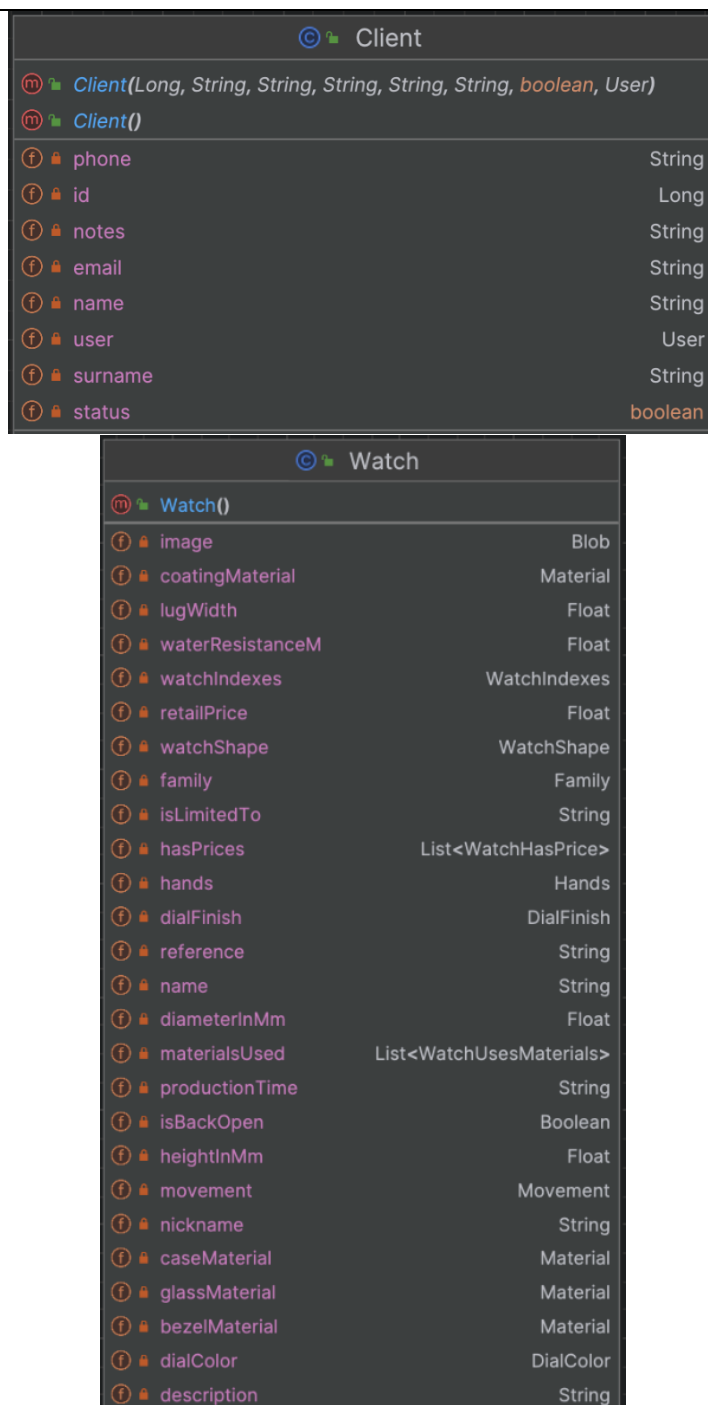
Nel design si può notare che abbiamo scelto una paletta di colori molto basilari, ma che combinati possono risultare molto professionali e moderni. Sulla pagina home si trova la ricerca degli orologi, la lista degli orologi e i preferiti. Nella navbar si possono vedere le altre pagine che verranno implementate.



3.4 Design procedurale

Diagramma delle classi, solo i model principali.





8 Diagramma Classi

4 Implementazione

4.1.1 MySQL

Per installare MySQL sul server bisogna eseguire i seguenti comandi da terminale:

```
sudo apt update
sudo apt install mysql-server
sudo systemctl start mysql.service
```

Dopo l'esecuzione di questi comandi bisognerà cambiare la password dell'utente root.

4.1.2 Node

Si è deciso di installare Node.js con NVM anziché utilizzare APT, poiché quest'ultimo non installa necessariamente l'ultima versione. Inoltre, considerando l'ampio utilizzo di librerie, sarebbe facile cambiare la versione di Node.js nel caso in cui certe librerie non fossero compatibili con la versione attuale.

Per l'installazione bisogna eseguire i seguenti comandi da terminale:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
source ~/.bashrc
```

Dopodichè si avrà la possibilità di scegliere la versione di Node con i seguenti comandi da terminale:

```
nvm list-remote
nvm install vXX.XX.X
nvm use vXX.XX.X.
```

4.1.3 Java

Per installare Java 21 si deve eseguire il seguente comando da terminale:

```
sudo apt install openjdk-21-jdk
```

4.1.4 Maven

Per installare Maven basta eseguire il seguente comando da terminale:

```
sudo apt install mvn
```

Dopo l'installazione è necessario impostare il proxy, altrimenti non sarà possibile scaricare le librerie usate. Per impostare il proxy bisogna modificare il file nella home dell'utente, al seguente percorso ".m2/settings.xml":

```
<settings>
  <proxies>
    <proxy>
      <id>proxy</id>
      <active>true</active>
      <protocol>http</protocol>
      <host>10.20.5.51</host>;
      <port>8888</port>
      <username></username>
      <password></password>
      <nonProxyHosts>localhost|127.0.0.1</nonProxyHosts>;
    </proxy>
  </proxies>
</settings>
```

4.1.5 Python3

Per installare Python3, che di solito è già installato, bisogna eseguire il seguente comando da terminale:

```
sudo apt install python3
```

Per installare Pip3, che serve a installare le librerie di Python, bisogna eseguire il seguente comando da terminale:

```
apt install python3-pip
```

4.1.6 Servizi

Per creare i servizi bisogna creare prima i file con estensione service.

nebulaWatchesAPI.service:

```
[Unit]
Description=nebulaWatchesAPI service
After=mysql.service
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=always
RestartSec=1
User=administrator
ExecStart=java -jar
/home/administrator/NebulaWatches/5_Applicativo/nebulaWatchesAPI/target/nebulaWatchesAPI-0.0.1-SNAPSHOT.jar

[Install]
WantedBy=multi-user.target
```

nebulaWatchesFrontend.service:

```
[Unit]
Description=nebulaWatchesFrontend service
After=nebulaWatchesFrontend.service
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=always
RestartSec=1
User=administrator
WorkingDirectory=/home/administrator/NebulaWatches/5 Applicativo/client
ExecStart=/usr/bin/npm run serve

[Install]
WantedBy=multi-user.target
```

nebulaWatchesChatbot.service:

```
[Unit]
Description=nebulaWatchesChatbot service
After=mysqld.service
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=always
RestartSec=1
User=administrator
ExecStart=/usr/bin/python3 /home/administrator/NebulaWatches/5 Applicativo/Chatbot/server.py

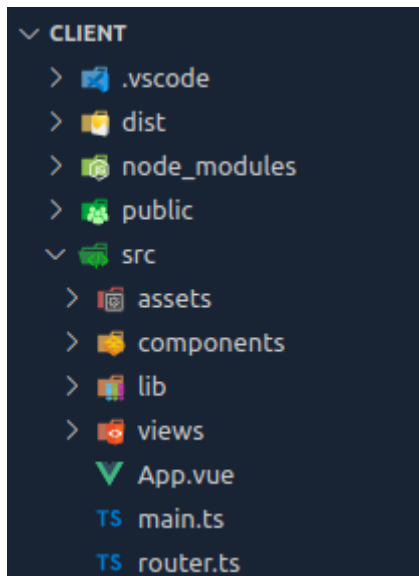
[Install]
WantedBy=multi-user.target
```

Dopo aver creato questi file è necessario spostarli in “/etc/systemd/system”, per poi abilitarli e farli partire con i seguenti comandi da terminale:

```
sudo systemctl enable nebulaWatchesAPI
sudo systemctl enable nebulaWatchesFrontend
sudo systemctl enable nebulaWatchesChatbot
sudo systemctl start nebulaWatchesAPI
sudo systemctl start nebulaWatchesFrontend
sudo systemctl start nebulaWatchesChatbot
```


4.2 Frontend

Il frontend è sviluppato con Vue JS usando la sintassi di script setup e composition api. Per la parte grafica abbiamo usato Tailwind in combinazione con Shadcn Vue. Abbiamo strutturato le view utilizzando componenti creati da noi e usando quelli di Shadcn Vue.



9 Struttura Frontend

4.2.1 Struttura View

Una view è strutturata da due parti principali, la prima è definita tra il tag `<template>`, la seconda tra il tag `<script setup>`.

La prima contiene tutta la parte grafica della pagina (comparabile a HTML e CSS), la seconda contiene tutta la logica, sia della pagina ma anche tutti i metodi che utilizzano l'API del backend per elaborare o prendere dati dal database (comparabile a Javascript).

Una view può anche contenere dei componenti, sia personali che da librerie, esse possono venir semplicemente usate importando il componente:

```
<template>
  <Alert variant="destructive">
    <AlertCircle class="w-4 h-4" />
    <AlertTitle>Error</AlertTitle>
    <AlertDescription>Alert!</AlertDescription>
  </Alert>
</template>

<script setup>
  import { Alert, AlertDescription, AlertTitle } from '@components/ui/alert'
</script>
```

4.2.2 Codice

In Vue JS è possibile usare “ref” e v-model per creare delle variabili o costanti che cambiano valore dinamicamente quando l’input a cui sono collegati cambiano valore, come in questo esempio quando viene inserito qualcosa nell’input, automaticamente il valore si trova nella variabile searchContent.

```
const searchContent = ref();

<Input @keyup="searchUser()" v-model="searchContent" type="text" placeholder="Search for users" />
```

Per fare delle richieste all’API del backend, è possibile usare Axios, come in questo esempio dove viene effettuata una richiesta POST a “/v1/admin/searchUser”, e come corpo della richiesta viene passato il contenuto della ricerca. Nell’header della richiesta viene inserito il token di autenticazione, se questo non fosse valido il server risponde con il codice 403 (forbidden).

Se il server risponde qualcosa, questo viene salvato nella response, che dopo può essere utilizzata per visualizzare i dati all’utente:

```
const response = await axios.post(`${apiServerAddress}/v1/admin/searchUser`, searchCt, {
  headers: {
    Authorization: 'Bearer ' + localStorage.getItem('token')
  },
});
```

La gestione degli errori causati da input errati dell’utente viene gestita con degli Alert, che a dipendenza dell’errore vengono visualizzati.

In questo esempio, viene visualizzato il messaggio “Failed, email is already used!”, se si cerca di fare la registrazione con un’e-mail già usata.

```
<Alert variant="destructive" v-if="emailUsed">
  <AlertCircle class="w-4 h-4" />
  <AlertTitle>Error</AlertTitle>
  <AlertDescription>Failed, email is already used! </AlertDescription>
</Alert>

const emailUsed = ref(false);
if(await isEmailUsed(newUser.email)){
  emailUsed.value = true;
}
```

In questo codice viene preso un’immagine casuale da un brand da visualizzare nelle carte degli orologi, questo viene utilizzato per prendere le immagini per la home, per la pagina dei brand e delle family. Viene eseguita una chiamata GET all’endpoint “/v1/brands/brandName/rndImage” in cui brandName corrisponde al nome del brand per cui si riceve un’immagine casuale. Se la richiesta va a buon fine, la risposta viene convertita in un’immagine Base64 utilizzando il metodo btoa() su un array di byte restituito dalla risposta.

```
async function fetchRandomWatchFromBrandImage() {
  try {
    const endpoint = `${apiServerAddress}/v1/brands/` + props.brand.name + '/rndimage'
    const response = await axios.get(endpoint, {
      responseType: 'arraybuffer',
      headers: {
        Authorization: 'Bearer ' + localStorage.getItem('token')
      },
    });
    const imageBase64 = btoa(new Uint8Array(response.data).reduce((data, byte) => data + String.fromCharCode(byte), ''));
    randomWatchFromBrandImage.value = `data:${response.headers['content-type'];base64,${imageBase64}`;
  } catch (error) {
    console.error('Failed to fetch image.');
```

In App.vue, che viene eseguito ogni volta che si cambia pagina, viene eseguito il seguente controllo. Questo serve per capire se il token dell'utente è già scaduto (il token scade dopo 24 ore dalla creazione) o addirittura non è presente, se questo dovesse essere il caso, l'utente viene reindirizzato sull'interfaccia del login. Questo permette di evitare problemi quando un utente cerca di fare delle richieste al server, ma questo non risponde più dato che il token non è più valido.

```
if (isTokenExpired()) {
  router.push('/login');
}

function isTokenExpired() {
  const token = localStorage.getItem('token');
  if (!token) return true;

  const tokenExp = JSON.parse(atob(token.split('.')[1])).exp * 1000;
  return tokenExp < Date.now();
}
```

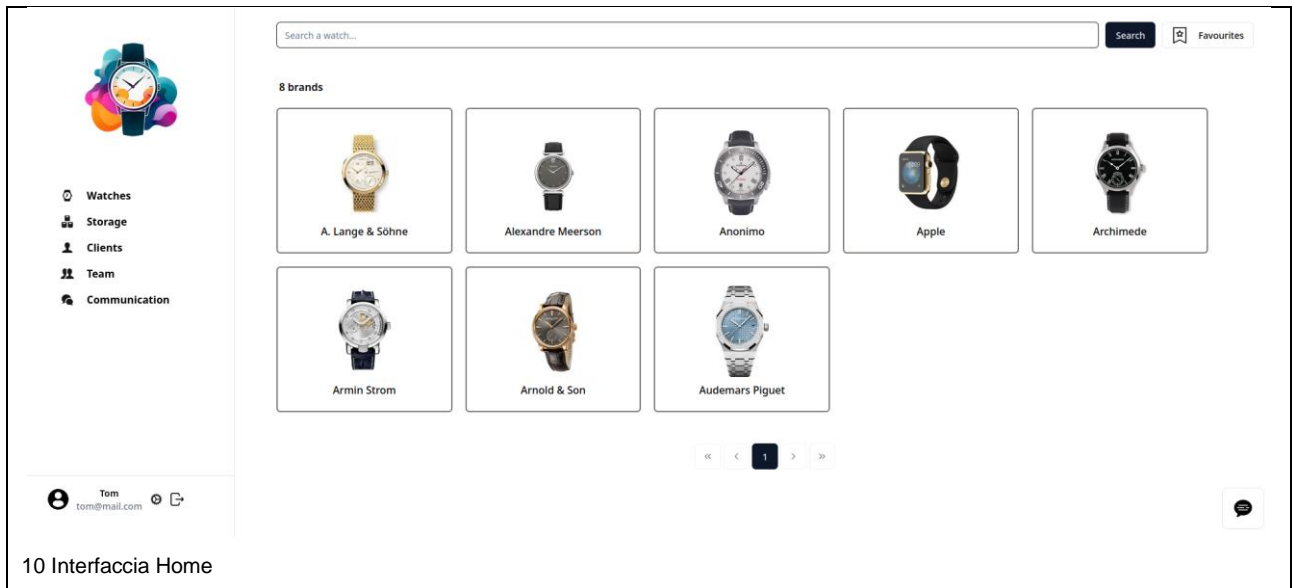
Un altro controllo che viene eseguito prima di ogni cambiamento di pagina è se l'utente sta cercando di accedere alla pagina d'admin e viene controllato se esso ha i permessi necessari per farlo. Se non si sta cercando di accedere alla pagina d'admin, il cambiamento di pagina viene lasciato passare senza controlli tramite il metodo next.

```
router.beforeEach(async (to, from, next) => {
  if (to.path === '/admin') {
    const token = localStorage.getItem('token');
    if (!token) {
      next(false);
      router.push('/login');
      return;
    }
    const parts = token.split('.');
    const payload = JSON.parse(atob(parts[1]));
    const email = payload.sub;
    await getRole(email);
    if (role.value !== "ADMIN") {
      next(false);
      router.push('/');
      return;
    }
  }
  next();
});
```

4.2.3 Interfaccia

Per poter confrontare il design iniziale e il risultato finale delle interfacce grafiche di seguito viene mostrato il risultato finale della pagina home.

Come si può notare i colori sono rimasti li stessi, a parte il viola che abbiamo deciso di rimpiazzare con il nero. Inoltre, la disposizione di alcuni pulsanti, come quello dei preferiti, è stato leggermente cambiato per comodità e praticità.



4.3 Backend

Il backend è principalmente basato su Java Spring Boot. Tuttavia, abbiamo sviluppato il chatbot in Python per ragioni di comodità e perché le librerie utilizzate non erano completamente compatibili con Java.



I file sono organizzati in diversi pacchetti, seguendo le best practice di Java Spring. Ogni pacchetto rappresenta una funzionalità specifica del sistema, come ad esempio 'watches' o 'security'.

All'interno di ciascun pacchetto, troviamo ulteriori suddivisioni:

- **config:** contiene i file di configurazione per l'applicazione Spring.
- **controller:** ospita i controller Spring responsabili della gestione delle richieste HTTP.
- **dto:** qui sono collocati i Data Transfer Object (DTO), utilizzati per semplificare il trasferimento dei dati tra il client e il server.
- **exception:** raccoglie le eccezioni personalizzate, come ad esempio 'WatchNotFoundException', utili per gestire situazioni anomale durante l'esecuzione dell'applicazione.
- **model:** contiene i modelli dati, rappresentati tramite classi Java, utilizzati dal sistema per rappresentare le entità.
- **repository:** include le repository Spring Data, che offrono un'interfaccia per l'interazione con il database. All'interno di queste repository, vengono definite le query personalizzate per l'accesso ai dati, ad esempio quelle per la ricerca degli orologi.
- **service:** contiene i servizi Spring, che implementano la logica di business dell'applicazione. Spesso, i servizi fanno uso delle repository per accedere ai dati e sono utilizzati dai controller per soddisfare le richieste degli utenti.

Per la documentazione delle API del backend di NebulaWatches, consultare il file *Documentazione_API.docx*.

4.3.1 Security

Il login dell'applicativo funziona con JWT (JSON Web Token), il quale viene generato al login di un utente e scade dopo 24 ore. Una volta autenticato, il token JWT può essere incluso nelle richieste successive fatte dal client al server. Il server può verificare la validità del token JWT e autorizzare l'accesso alle risorse in base alle informazioni contenute nel token stesso, per esempio in base al ruolo (User o Admin). Questo migliora la sicurezza e inoltre questo metodo offre una scalabilità molto ampia dato che facilita l'autenticazione tra più server. Inoltre, in questo package si trovano tutti i metodi collegati all'admin.

4.3.1.1 Config

Nella cartella config si trovano tutte le classi e metodi che servono a configurare in modo corretto il login con JWT e a definire a quali pagine un utente può accedere.

Nella classe SecurityConfig viene impostato una whitelist degli URL che sono accessibili senza login, perciò tutte le pagine che hanno a che fare con l'autenticazione, nel nostro caso "/auth".

Nella classe JwtAuthenticationFilter viene filtrato il token, e tramite metodi presenti nei services, verifica se il token fornito è valido.

Nella Classe AppConfig sono definiti vari metodi che servono a facilitare l'uso dei metodi di autenticazione nelle altre classi.

4.3.1.2 Codice

Quando un utente si registra, o viene registrato tramite l'admin, esso riceve un'e-mail con un codice che serve a verificare il nuovo account. Ecco il metodo sendEmail di EmailService, che permette di mandare un'e-mail contenente il codice di verifica all'utente tramite l'API di Mailjet. Il contenuto dell'e-mail è scritto in HTML per poter applicare uno stile personalizzato ad essa. Per poter mandare un'e-mail bisogna impostare l'indirizzo di destinazione e di partenza, il contenuto e un oggetto.

```
public void sendEmail(String email, int code) throws MailjetException {
    MailjetClient client;
    MailjetRequest request;
    MailjetResponse response;
    client = new MailjetClient("", "");
    request = new MailjetRequest(Emailv31.resource)
        .property(Emailv31.MESSAGES, new JSONArray()
            .put(new JSONObject()
                .put(Emailv31.Message.FROM, new JSONObject()
                    .put("Email", "nebulawatchesproject@gmail.com")
                    .put("Name", "nebulawatchesproject@gmail.com"))
                .put(Emailv31.Message.TO, new JSONArray()
                    .put(new JSONObject()
                        .put("Email", email)
                        .put("Name", email)))
                .put(Emailv31.Message.SUBJECT, "LogIn in NebulaWatches")
                .put(Emailv31.Message.HTMLPART,
                    "<title>Benvenuto su NebulaWatches</title>"
                )));
    response = client.post(request);
    System.out.println(response.getStatus());
    System.out.println(response.getData());
}
```

Il metodo authenticate in AuthenticationService serve a controllare se le credenziali inserite dall'utente al login sono corrette. Esso controlla se l'email è valida e che l'account non sia disabilitato. Dopodiché, usando l'AuthenticationManager di Java Spring viene controllato se l'email e password inserita sono corrette. Infine, genera un token JWT per l'utente autenticato utilizzando jwtService e lo ritorna. Se i valori inseriti non dovessero essere corretti, viene ritornato null, che poi viene gestito più tardi a visualizzare un errore adeguato all'utente.

```
public AuthenticationResponse authenticate(AuthenticationRequest request) {
    if (InputUtils.isEmailValid(request.getEmail()) &&
        !repository.isArchived(request.getEmail().get())) {
        authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(
                InputUtils.testInput(request.getEmail()),
                InputUtils.testInput(request.getPassword())
            )
        );
        var user = repository.findByEmail(InputUtils.testInput(request.getEmail())).orElseThrow();
        var jwtToken = jwtService.generateToken(user);
        return AuthenticationResponse.builder()
            .token(jwtToken)
            .build();
    }
    return null;
}
```

Questa parte di codice presente in AdminController serve a controllare che chi esegue questi metodi è veramente un Admin, questo viene fatto estraendo l'e-mail dell'utente dal JWT e controllandone il ruolo nel database. Se l'utente è Admin viene poi chiamato il metodo getAllUsers, che serve a prendere tutti gli utenti dell'applicativo.

```
String token = headers.getFirst("Authorization");
if(token != null && adminService.isAdminByToken(token)){
    return adminService.getAllUsers();
}else{
    return null;
}
```

4.3.2 Storage

Il package storage contiene tutto quello che ha a che fare con gli orologi preferiti di un utente e gli orologi che possiede o ha venduto. In questo package sono presenti molti metodi CRUD, molto lunghi a causa di controllo di input e non molto interessanti, per questo motivo non verranno presentati qui. Può essere interessante però sapere che, se si aggiunge un orologio allo storage che ha i dati identici a uno già presente, questi vengono messi insieme.

Questo metodo in StorageService serve a ritornare il numero di acquisti fatti da un singolo cliente in uno specifico mese, questo è necessario per un grafico di statistica. Per prima cosa si delimita il mese e poi viene fatta una richiesta al database tramite lo storageRepository. Un fatto interessante è che tutte le query per il nostro applicativo sono scritte con JPA, ma sumQuantityByClientMonth no, dato che ha problemi a fare i confronti con le date.

```
public int getWatchesOwnedByClientMonth(Long id, int month){
    LocalDate l1 = getStartOfMonth(month);
    LocalDate l2 = getEndOfMonth(month);
    Optional<Integer> opt = storageRepository.sumQuantityByClientMonth(id, l1, l2, "Sold");
    return opt.orElse(0);
}
```

Questo metodo in CustomWatchService serve a controllare se il file inserito come immagine di un orologio personalizzato dall'utente è valido, se è valido ritorna i byte dell'immagine per poterli salvare nel database tramite il repository. Se invece non è valido viene generata un'eccezione che poi mostrerà un messaggio di errore all'utente.

```
public byte[] setImage(MultipartFile imageFile) throws IOException {
    if (imageFile != null && !imageFile.isEmpty()) {
        if (!InputUtils.isValidImageType(imageFile)) {
            throw new IllegalArgumentException("Invalid file type. Please upload an image file.");
        }
        return imageFile.getBytes();
    } else {
        throw new IllegalArgumentException("Image file is required.");
    }
}
```

4.3.3 Team

Il package team contiene tutto quello che ha a che fare con i membri del team, che vendono gli orologi del nostro negozio. Anche in qui la maggior parte dei metodi sono CRUD, perciò non è presente molto codice interessante, ma farò vedere comunque una volta un'aggiunta di un nuovo membro del team. I valori vengono presi dalla request che contiene i valori inseriti dall'utente passati dal frontend, vengono sanitizzati tramite il metodo testInput e poi vengono inseriti nel nuovo team. Se l'email del utente associato non è corretta, e perciò non è possibile trovare uno user, viene lanciato un errore.

```
public void saveTeamMember(TeamRequest request){
    Team team = new Team();
    team.setEmail(InputUtils.testInput(request.getEmail()));
    team.setName(InputUtils.testInput(request.getName()));
    team.setSurname(InputUtils.testInput(request.getSurname()));
    team.setPhone(InputUtils.testInput(request.getPhone()));
    team.setRole(InputUtils.testInput(request.getRole()));
    if(InputUtils.isValidEmail(request.getUserEmail())){
        User user = userRepository.findByEmail(InputUtils.testInput(request.getUserEmail()))
            .orElseThrow(() -> new IllegalArgumentException("User not found"));
        team.setUser(user);
    }else{
        throw new IllegalArgumentException("User email not valid!");
    }
    teamRepository.save(team);
}
```

4.3.4 Client

Il package client contiene tutto quello che ha a che fare con i clienti, essi vengono inseriti come acquirenti degli orologi nello storage. In questo package sono solo contenuti pochi metodi e classi, le quali non hanno contenuto interessante.

4.3.5 Watches

Il package "Watches" contiene tutto ciò che riguarda gli orologi. Questo è il package più corposo, con moltissimi models, services, controllers e repositories.

Questo metodo in WatchesService controlla i parametri in entrata, e se essi sono errati solleva un'eccezione, altrimenti imposta una paginazione per la ricerca e tramite watchRepository prende le pagine di orologi cercati dalla query. Vengono ritornati in pagine per rendere l'applicativo più performante dato che deve prendere solo una pagina di 20 orologi e non tutti i risultati.

```
public Page<Watch> getWatchesBySearchQueryPage(String query, int page, int pageLength, String
sortBy) {
    if (page < 0 || pageLength <= 0) {
        throw new IllegalArgumentException("Invalid page or length parameters");
    }
    Sort.Direction sortDirection = Sort.Direction.ASC;

    Pageable paging = PageRequest.of(page, pageLength, sortDirection, sortBy);

    return watchRepository.findBySearchQuery(query, paging);
}
```


Nel package Watches è inoltre presente un DTO (Data Transfer Object), utilizzato per gestire le relazioni molti a molti presenti tra i diversi model degli orologi e a trasmettere in modo completo e strutturato le informazioni su un orologio all'interno dell'applicazione.

```
if (this.materialsUsed != null) {
    List<String> materialNames = this.materialsUsed.stream()
        .map(wum -> wum.getMaterial().getName())
        .collect(Collectors.toList());
    watchDTO.setMaterialsUsedNames(materialNames);
}

if (this.hasPrices != null) {
    List<Float> prices = new ArrayList<>();
    List<Date> dates = new ArrayList<>();
    for (WatchHasPrice price : this.hasPrices) {
        prices.add(price.getPrice());
        dates.add(price.getDate());
    }
    watchDTO.setPrices(prices);
    watchDTO.setDates(dates);
}
```

4.3.6 Utils

Il package “Utils” contiene una classe contenente metodi statici di utility.

Il primo è testInput, il quale prende una stringa passata come parametro e la sanitizza per evitare attacchi di sql injection, per prima cosa rimuove gli spazi inutili con trim, poi rimpiazza i backslash e trasforma i caratteri speciali in stringhe codificate, così da non poter recar danno. Alla fine, ritorna la stringa sanitizzata.

```
public static String testInput(String data) {
    String sanitizedData = data.trim();
    sanitizedData = sanitizedData.replaceAll("\\\\", "");
    sanitizedData = StringEscapeUtils.escapeHtml4(sanitizedData);
    return sanitizedData;
}
```

Il secondo metodo di utility è usato per verificare se una stringa rispetta un certo pattern, in questo caso quello di un'e-mail. Per essere valida la stringa deve iniziare con uno o più caratteri alfanumerici, seguiti da un punto e altri caratteri alfanumerici. Dopo la chiocciola (@) deve esserci uno o più caratteri alfanumerici, seguiti da un punto e conclusi con caratteri alfanumerici, per esempio, un'e-mail valida potrebbe essere nebula.watches@mail.com.

```
public static boolean isEmailValid(String email) {
    String regex = "[a-zA-Z0-9_+&*-]+(?:\\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z-]{2,7}$";
    Pattern pattern = Pattern.compile(regex);
    Matcher matcher = pattern.matcher(email);
    return matcher.matches();
}
```

L'ultimo metodo di utility serve a verificare se un'immagine passata come MultipartFile (classe fornita da Spring che rappresenta un file caricato attraverso un input HTML), è di un formato supportato (jpeg, jpg, gif e png).

```
public static boolean isValidImageType(MultipartFile file) {
    List<String> allowedTypes = Arrays.asList("image/jpeg", "image/png", "image/gif", "image/jpg");
    return allowedTypes.contains(file.getContentType());
}
```

4.3.7 Chatbot

Il Chatbot è sviluppato in Python usando Langchain, Lanchain serve a collegare applicazioni con gli LLM. Nel nostro caso viene creato un “Agent”, questo in base ad una query capisce che “Tool” utilizzare, questi servono a dare più contesto (RAG) o fare azioni ad un “Agent”, possono ritornare tutto quello che vogliamo, ad esempio visto che GPT 3.5 non dispone di informazioni in tempo reale, possiamo creare un tool chiamato “GetWeatherData”, ma nel nostro caso i tool sono due e servono a ritornare informazioni aggiuntive.

- **watch_database_tool**: comunica con il database, crea delle query da eseguire sul database per retribuire i dati necessari, con quei dati genera dopo una risposta.
- **ask_chat_gpt**: comunica direttamente con l’API di OpenAI senza step aggiuntivi.

4.3.7.1 API

Come framework per l’API è stato scelto FastAPI, per la sua semplicità, il codice è tutto in server.py, contiene semplicemente l’endpoint “/ask_bot” dove si comunica con “Agent”, poi ci sono anche le configurazioni del CORS e l’autenticazione con il JWT.

```
AUTH_SERVER = 'http://10.20.4.113:64321'
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["GET"],
    allow_headers=["*"],
)
```

Qua viene istanziato l’oggetto FastAPI e vengono fornite le impostazioni del CORS, non c’è bisogno di niente di speciale, viene inoltre fornito l’ip del nostro server per l’autorizzazione.

```
def is_token_valid(jwt):
    r_url = f"{AUTH_SERVER}/auth/isTokenValid"
    params = {'jwt': jwt}
    r = requests.get(r_url, params=params)

    return r.text

async def authentication_middleware(request, call_next):
    token = request.headers.get("jwt")

    if not is_token_valid(token):
        print(is_token_valid(token))
        raise HTTPException(status_code=401, detail="Invalid JWT token")

    response = await call_next(request)
    return response

app.middleware("http")(authentication_middleware)
```

Inoltre è presente un metodo di middleware, questo serve a controllare che ci sia un JWT valido, per non disporre degli endpoint alle persone non autenticate, il token viene controllato dal backend. Infine viene aggiunto il middleware per tutti gli endpoint.

```
agent_map = {}

@app.get('/ask_bot')
async def ask_bot(query: str, jwt: str = Header(...)):
    email = is_token_valid(jwt)
    if jwt not in agent_map:
        agent_map[jwt] = Agent(email)

    result = agent_map[jwt].ask(query)
    return {'result': result}
```

Il contenuto importante è che istanzia un “Agent” per ogni utente, se un utente ha già utilizzato l’endpoint, non viene istanziato un altro oggetto, su questa cosa si può giocare molto e al momento è molto minimale.

4.3.7.2 Agent

Un “Agent” come già detto in precedenza serve a fare ragionamenti su che tool utilizzare e come mettere assieme le informazioni che riceve dai tool.

```
load_dotenv()

@tool
def watch_database_tool(query: str, user_email: str) -> str:
    """A tool to query a MySQL database about personal info, user, watches, clients, storage,
    purchases, prices and other related information, and get natural language responses. Do not use for
    any type of suggestions"""

    chatbot = Chatbot(user_email)
    chatbot_answer = chatbot.ask_question(query)
    return chatbot_answer

@tool
def ask_chat_gpt(query: str) -> str:
    """A function to query ChatGPT for any type of suggestions about watches, lifestyle, etc."""

    llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0.5)
    response = llm.invoke(query)
    return response
```

All’inizio viene caricato il file .env contenente informazioni sensibili come, la chiave dell’API di OpenAI e le credenziali del database del progetto.

Dopodiché vengono creati i tool, i tool devono avere nomi intuitivi ed è obbligatorio che abbiano la docstring, ovvero il commento prima del codice, se non viene disposto il programma ritornerà un errore, perché altrimenti un “Agent” non riesce a capire bene le funzionalità del “Tool” e se gli serve o no.

Il primo “Tool” serve a ritornare informazioni dal database del progetto, prende come parametro l’email dell’utente che ha fatto una query ad un “Agent”, l’email dell’utente viene ricavata dal JWT.

Il secondo “Tool” fa semplicemente una richiesta all’API di OpenAi con una query generata da un “Agent”, un “Agent” capisce da solo cosa usare come parametro dei “Tool”.

```
class Agent:
    def __init__(self, user_email):
        self.user_email = user_email
        self.initialize_agent()

    def initialize_agent(self):
        llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0.5)
        tools = [watch_database_tool, ask_chat_gpt]
        prompt = ChatPromptTemplate.from_messages(
            [
                (
                    "system",
                    f"""You are very powerful assistant, called Gio, but don't know current events,
you can use tools at a time, user email is {self.user_email} so you only have access to this user,
do not query about data relating other users

Additional rules:
- Always respond in the language you were asked.
- If the users asks for a watch collection, use the ChatGptTool, with a budget
use the entire budget, not only a part.
- Please do not repeat yourself.
""",
                ),
                ("user", "{input}"),
                MessagesPlaceholder(variable_name="agent_scratchpad"),
            ]
        )
        llm_with_tools = llm.bind_tools(tools)

        agent = (
            {
                "input": lambda x: x["input"],
                "agent scratchpad": lambda x: format_to_openai_tool_messages(
                    x["intermediate_steps"]
                ),
            }
            | prompt
            | llm_with_tools
            | OpenAIToolsAgentOutputParser()
        )

        self.agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True)

    def ask(self, query):
        return list(self.agent_executor.stream({"input": query}))[-1]['output']
```

Dopodiché si crea la classe del "Agent", anche qui si passa l'email precedentemente presa dal JWT. È interessante notare che si ha la scelta del modello di LLM di OpenAI, nel caso specifico gpt-3.5-turbo è stato selezionato per la sua velocità e perché è recente. Insieme a ciò si specifica la temperatura da 0 a 1. Con 0 il modello risponde teoricamente sempre alla stessa maniera, mentre verso 1 diventa più creativo e a volte può dare informazioni diverse tra loro. È stata scelta la temperatura 0 poiché non si desidera un comportamento creativo, visto che ciò implica un rischio nella correttezza delle risposte.

Dopo la scelta del modello, è possibile specificare un prompt. Questo prompt serve a modificare certi pensieri di un "Agent". Se si è molto precisi ed espliciti, un "Agent" farà i ragionamenti e le azioni desiderate. Nel caso specifico, si specificano delle piccole cose che sono state notate non funzionare sempre correttamente durante la fase di test.

Successivamente viene creata una "Chain", che include vari step eseguiti, il prompt, un modello di LLM, eccetera. Da questa "Chain" viene creato un "AgentExecutor" pronto a rispondere alle query.

4.3.7.3 Tool Database

Infine c'è il "Tool" che comunica con il database del progetto, questo come detto precedentemente, in base ad una query genera una query MySQL che poi viene eseguita, dopo l'esecuzione viene generata una risposta basata su quello che ritorna la query MySQL.

```
class Chatbot:
    def __init__(self, user_email):
        load_dotenv()

        self.user_email = user_email
        print('CHATBOTO ', user_email)

        self.DB_USER = os.getenv('DB_USER')
        self.DB_PASSWORD = os.getenv('DB_PASSWORD')
        uri =
f'mysql+mysqlconnector://{self.DB_USER}:{self.DB_PASSWORD}@localhost:3306/NebulaWatches'

        self.db = SQLiteDatabase.from_uri(uri)
        self.llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)

        self.define_sql_chain()
        self.define_full_chain()

    def get_schema(self, db):
        schema = self.db.get_table_info()
        return schema

    def run_query(self, query):
        return self.db.run(query)
```

All'inizio si definiscono le credenziali del database del progetto, dopodiché si definiscono le "Chain", quella che crea la query e quella che comprende tutto.

Sono presenti anche le funzioni per prendere le informazioni dello "Schema" del database e per eseguire le query MySQL.

```
def define_sql_chain(self):
    template = """As a MySQL expert with access to a database containing information about
watches, clients, storage, purchases, and related data, your task is to construct SQL queries based
on the table schema provided below to answer user questions.

    USER_QUERYING_EMAIL: {user_email}

    You are only permitted to access data associated with the USER_QUERYING_EMAIL user email.
Any attempt to retrieve information about other users must result in returning NULL.

    Ensure that all data accessed belongs to the user identified by USER_QUERYING_EMAIL. If not,
return "SELECT NULL;".

    For instance, if someone queries about 'gino@gmail.com' storage but the USER_QUERYING_EMAIL
is different, do not disclose 'gino@gmail.com' information. Instead, return "SELECT NULL;".

    Another example, if the users asks something like "What watches have we sold", only give
the user identified by USER_QUERYING_EMAIL data, not all users.

    Certain data such as clients, teams, storage, and sold storage are linked to specific users.
Your queries must retrieve data where the user email matches USER_QUERYING_EMAIL, or return "SELECT
NULL;".

    Avoid performing any DML statements (INSERT, UPDATE, DELETE, DROP, etc.) on the database.

    If a question seems unrelated to the database, return "SELECT NULL;".

    Use only the provided tools and the information returned by them to construct your query.
Additionally, refrain from querying image or BLOB columns.
```

When constructing queries, prioritize using names over IDs or technical identifiers. Always include the watch reference when necessary.
Never use ids, use always names if possible.

```
{schema}

Question: {question}
SQL Query:
"""
prompt = ChatPromptTemplate.from_template(template)

self.sql_chain = (
    RunnablePassthrough.assign(schema=self.get_schema)
    | RunnablePassthrough.assign(user_email=lambda vars: self.user_email)
    | prompt
    | self.llm.bind(stop=["\nSQLResult:"])
    | StrOutputParser()
)
```

Come si può vedere la “Chain” non è molto difficile da quello che si è visto prima, l’unica cosa differente è che il prompt è molto lungo, perché si devono definire le cose a cui la “Chain” può accedere, ad esempio non diamo il permesso di eseguire query DML, che non può accedere ai dati di altri utenti, in certi casi è stato definito di ritornare “RETURN NULL;” in certe situazioni non si presentava il comportamento voluto da noi.

```
def define_full_chain(self):
    template = """Based on the table schema below, question, sql query, and sql response, write
a natural language response.

    If you don't have enough information in the database to answer the question, you can provide
suggestions or additional context as a watch expert, but make it clear that the information is not
coming from the database.

    If the question is not related to watches, clients, storage, purchases, or any other data in
the database, reply with "As a watch expert, this question goes beyond my knowledge. I would suggest
consulting [relevant resource or expert]."
```

When suggesting a watch, always provide the watch reference.
Format the response in a way that is easily readable for humans

```
{schema}

Question: {question}
SQL Query: {query}
SQL Response: {response}"""
prompt_response = ChatPromptTemplate.from_template(template)

self.full_chain = (
    RunnablePassthrough.assign(query=self.sql_chain).assign(
        schema=self.get_schema,
        response=lambda vars: self.run_query(vars["query"]),
    )
    | prompt_response
    | self.llm
)
```

La “full_chain” unisce la “sql_chain” e ritorna una risposta leggibile da un essere umano, in base al contenuto del risultato della query MySQL, gestisce anche l’evenienza che il risultato della query MySQL sia nullo.

```
def ask_question(self, question):
    response = self.full_chain.invoke({"question": question})
    return response.content
```

Infine viene invocata la “full_chain” quando si vogliono fare domande al database.

5 Test

5.1 Protocollo di test

Per eseguire i test dove è necessario un login, utilizzare il seguente account: email: test@progetto.com, password: Admin\$00

| | | | |
|--------------------------|---|--------------|----------------------|
| Test Case: | TC-001 | Nome: | Registrazione utente |
| Riferimento: | REQ-002 | | |
| Descrizione: | Passando username, email, password ci si può registrare. | | |
| Prerequisiti: | Inserire nella barra di ricerca: http://10.20.4.113:8080/register | | |
| Procedura: | <ol style="list-style-type: none"> 1. Inserire i valori richiesti (username, email, password) 2. Premere Register | | |
| Risultati attesi: | Ci si trova nella pagina di inserimento del Pin di verifica. | | |

| | | | |
|--------------------------|---|--------------|----------------|
| Test Case: | TC-002 | Nome: | Verifica Email |
| Riferimento: | REQ-003 | | |
| Descrizione: | Verifica dell'e-mail di un nuovo account. | | |
| Prerequisiti: | Avere effettuato la registrazione, TC-001. Se non ci si trova nella pagina con il form di verifica, effettuare il login con le credenziali dell'account non verificato. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Controllare la mailbox dell'e-mail con cui ci si è registrati. 2. Inserire il codice dell'e-mail nel form di verifica. 3. Premere il pulsante. | | |
| Risultati attesi: | Si è autenticati nel sito e lo si può utilizzare. Al prossimo login non sarà più necessario fare una verifica. | | |

| | | | |
|--------------------------|--|--------------|--------------|
| Test Case: | TC-003 | Nome: | Login utente |
| Riferimento: | REQ-001 | | |
| Descrizione: | Passando e-mail, password, viene autenticato l'utente e viene generato un token di autenticazione | | |
| Prerequisiti: | Avere un utente registrato. Inserire nella barra di ricerca: http://10.20.4.113:8080 (bisogna prima eseguire il logout (TC-004) se si ha precedentemente eseguito TC-002). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Inserire i valori richiesti (e-mail, password), dell'account creato in TC-001 2. Premere Login | | |
| Risultati attesi: | Ci si trova nella pagina home dell'applicativo. | | |

| | | | |
|--------------------------|--|--------------|--------|
| Test Case: | TC-004 | Nome: | Logout |
| Riferimento: | REQ-001 | | |
| Descrizione: | Logout dall'applicativo | | |
| Prerequisiti: | Avere effettuato il login (TC-003) e trovarsi su http://10.20.4.113:8080 | | |
| Procedura: | 1. Schiacciare il pulsante della porta di uscita nella sidebar. | | |
| Risultati attesi: | Ci si ritrova nella pagina di login. | | |

| | | | |
|--------------------------|--|--------------|------------------|
| Test Case: | TC-005 | Nome: | Login con Google |
| Riferimento: | REQ-002 | | |
| Descrizione: | Login con account Google all'applicativo | | |
| Prerequisiti: | Andare nella pagina di login, http://10.20.4.113:8080 | | |
| Procedura: | <ol style="list-style-type: none"> 1. Schiacciare il pulsante del Login con Google 2. Inserire le proprie credenziali Google | | |
| Risultati attesi: | Ci si trova nella home dell'applicativo autenticato. | | |

| | | | |
|--------------------------|--|--------------|-------------------------|
| Test Case: | TC-006 | Nome: | Visualizzazione orologi |
| Riferimento: | REQ-007 | | |
| Descrizione: | Procedura per visualizzare orologi. | | |
| Prerequisiti: | Trovarsi nella homepage e aver effettuato il login. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Schiacciare A. Lange & Söhne (brand) 2. Schiacciare Lange 1 (family) 3. Schiacciare Lange 1 Yellow Gold (watch) | | |
| Risultati attesi: | Si visualizza l'orologio selezionato e tutti i suoi attributi | | |

| | | | |
|--------------------------|--|--------------|-------------------------|
| Test Case: | TC-007 | Nome: | Aggiungere ai preferiti |
| Riferimento: | REQ-007 | | |
| Descrizione: | Aggiungere un orologio ai preferiti. | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere su un brand, poi su una family e poi su un orologio. 2. Premere la stella per aggiungere ai preferiti. 3. Premere il pulsante Favourites. | | |
| Risultati attesi: | Si può visualizzare l'orologio appena aggiunto ai preferiti. | | |

| | | | |
|--------------------------|---|--------------|------------------------|
| Test Case: | TC-008 | Nome: | Togliere dai preferiti |
| Riferimento: | REQ-007 | | |
| Descrizione: | Rimuovere un orologio dai preferiti. | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. Avere un orologio nei preferiti (TC-007). | | |
| Procedura: | <ol style="list-style-type: none"> Andare in Favourites. Premere sull'orologio Premere sulla stella di colore nero per togliere dai preferiti. | | |
| Risultati attesi: | Nella pagina favourite non è più presente l'orologio. | | |

| | | | |
|--------------------------|---|--------------|---------------------------------|
| Test Case: | TC-009 | Nome: | Aggiungere allo storage "Owned" |
| Riferimento: | REQ-006 | | |
| Descrizione: | Aggiungere un orologio allo storage con stato "Owned" (di cui si è proprietari) | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. | | |
| Procedura: | <ol style="list-style-type: none"> Andare su un orologio Premere sul pulsante "Add to storage" Scegliere lo status "Owned" Inserire il resto dei dati Premere "Add to storage" | | |
| Risultati attesi: | Appare un messaggio di conferma che è stato aggiunto. Se si va nella pagina storage l'orologio è stato aggiunto. Se si preme sull'orologio nello storage si vedono le informazioni dell'orologio. | | |

| | | | |
|--------------------------|--|--------------|--------------------------------|
| Test Case: | TC-010 | Nome: | Aggiungere allo storage "Sold" |
| Riferimento: | REQ-006 | | |
| Descrizione: | Aggiungere un orologio allo storage con stato "Sold" (che si ha venduti). | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. Avere inserito un cliente e un membro del team (tab "Owned"). | | |
| Procedura: | <ol style="list-style-type: none"> Andare su un orologio Premere sul pulsante "Add to storage" Scegliere lo status "Sold" Inserire il resto dei dati Premere "Add to storage" | | |
| Risultati attesi: | Appare un messaggio di conferma che è stato aggiunto. Se si va nella pagina storage l'orologio è stato aggiunto (tab "Sold"). Se si preme sull'orologio nello storage si vedono le informazioni dell'orologio. | | |

| | | | |
|--------------------------|---|--------------|----------------------------------|
| Test Case: | TC-011 | Nome: | Eliminazione orologio da storage |
| Riferimento: | REQ-006 | | |
| Descrizione: | Eliminazione di un orologio dallo storage. | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. Avere almeno un orologio nello storage. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Andare in storage 2. Premere sull'icona del cestino. 3. Confermare | | |
| Risultati attesi: | L'orologio è stato eliminato dallo storage. | | |

| | | | |
|--------------------------|---|--------------|---------------------|
| Test Case: | TC-012 | Nome: | Paginazione orologi |
| Riferimento: | REQ-007 | | |
| Descrizione: | Cambiare pagina di orologi/brand/family. | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Andare in fondo alla pagina 2. Premere sul numero 2 (o sulla freccia) | | |
| Risultati attesi: | L'utente viene riportato in alto alla pagina e si trova sulla seconda pagina. | | |

| | | | |
|--------------------------|---|--------------|-----------------|
| Test Case: | TC-013 | Nome: | Ricerca orologi |
| Riferimento: | REQ-007 | | |
| Descrizione: | Fare la ricerca sugli orologi in base a brand, family, reference e descrizione. | | |
| Prerequisiti: | Avere eseguito il login e trovarsi nella home. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere nella barra di ricerca e digitare "Zeitwerk Duncan". | | |
| Risultati attesi: | Viene visualizzato un'orologio di nome "Zeitwerk Duncan Wang / Kidz Horizon" | | |

| | | | |
|--------------------------|--|--------------|-----------------------|
| Test Case: | TC-014 | Nome: | Cambio stato orologio |
| Riferimento: | REQ-006 | | |
| Descrizione: | Modifica dello stato di orologio da "Owned" a "Sold" | | |
| Prerequisiti: | Avere un utente registrato. Avere almeno un orologio nello storage con stato "Owned". Avere un cliente e un membro del team inserito nell'applicazione. | | |
| Procedura: | <ol style="list-style-type: none"> 1. In storage premere sull'icona della penna per modificare lo stato. 2. Inserire numero di orologi, il prezzo singolo di un orologio, il cliente, il membro del team che ha venduto l'orologio e la data di vendita. 3. Premere save changes. | | |
| Risultati attesi: | Se si hanno modificato tutti gli orologi di quell'orologio (quantità massima), l'orologio non è più presente in Owned, ma solo in Sold. Altrimenti si trova ancora in tutti e due, con le nuove quantità. | | |

| | | | |
|--------------------------|---|--------------|-----------------|
| Test Case: | TC-015 | Nome: | Prezzo orologio |
| Riferimento: | REQ-007 | | |
| Descrizione: | Visualizzazione il prezzo di un orologio | | |
| Prerequisiti: | Avere un utente registrato. | | |
| Procedura: | <ol style="list-style-type: none"> Andare su orologio con reference: 126900-0001 (tramite ricerca) Entrare sull'orologio e sotto al prezzo premere show more. | | |
| Risultati attesi: | Viene visualizzato il prezzo nel tempo (se ci sono dati disponibili). | | |
| Test Case: | TC-016 | Nome: | Admin panel |
| Riferimento: | REQ-010 | | |
| Descrizione: | Accesso all'admin panel. | | |
| Prerequisiti: | Avere un utente admin. Eseguire il login con le credenziali da admin: Email: adm@mail.com Password: Admin\$00 | | |
| Procedura: | <ol style="list-style-type: none"> Effettuare il login con questo utente | | |
| Risultati attesi: | Ci si trova nella pagina di amministrazione. | | |

| | | | |
|--------------------------|--|--------------|-------------------|
| Test Case: | TC-017 | Nome: | Aggiungere utente |
| Riferimento: | REQ-010 | | |
| Descrizione: | Aggiungere utente dal pannello admin. | | |
| Prerequisiti: | Trovare nella pagina di admin (TC-016). | | |
| Procedura: | <ol style="list-style-type: none"> Premere Add User Inserire dati validi Premere Save | | |
| Risultati attesi: | Il nuovo utente è stato aggiunto ed è visibile nella lista. | | |

| | | | |
|--------------------------|---|--------------|---------------|
| Test Case: | TC-018 | Nome: | Modifica User |
| Riferimento: | REQ-010 | | |
| Descrizione: | Modificare un utente dal pannello admin. | | |
| Prerequisiti: | Avere un utente nel sistema (TC-017). Trovare nella pagina admin. | | |
| Procedura: | <ol style="list-style-type: none"> Premere su Edit nella riga dell'utente aggiunto nel test case precedente (TC-017) Modificare i dati Salvare | | |
| Risultati attesi: | L'utente appare con i nuovi valori. | | |

| | | | |
|--------------------------|---|--------------|--------------|
| Test Case: | TC-019 | Nome: | Elimina User |
| Riferimento: | REQ-010 | | |
| Descrizione: | Eliminare un utente | | |
| Prerequisiti: | Avere un utente nel sistema (TC-017). Trovarsi nella pagina admin. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sul pulsante Delete sulla riga dell'utente aggiunto in TC-017 2. Confermare l'eliminazione. | | |
| Risultati attesi: | Utente eliminato (in realtà è archiviato per mantenere la coerenza dei dati). | | |

| | | | |
|--------------------------|--|--------------|----------------|
| Test Case: | TC-020 | Nome: | Ricerca Utente |
| Riferimento: | REQ-010 | | |
| Descrizione: | Ricerca utenti dal pannello admin. | | |
| Prerequisiti: | Avere un utente nel sistema (TC-017). Trovarsi nella pagina admin. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Scrivere nella barra di ricerca quello che si vuole cercare | | |
| Risultati attesi: | La lista di utenti si compone solo dagli utenti filtrati per la ricerca. | | |

| | | | |
|--------------------------|---|--------------|-------------------|
| Test Case: | TC-021 | Nome: | Paginazione admin |
| Riferimento: | REQ-010 | | |
| Descrizione: | Paginazione della pagina admin. | | |
| Prerequisiti: | Avere almeno 5 utenti nel sistema (TC-017). Trovarsi nella pagina admin. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Aggiungere un ulteriore utente (TC-017) 2. In fondo alla pagina premere "2" o sulla freccia | | |
| Risultati attesi: | Ci si ritrova nella seconda pagina di utenti. | | |

| | | | |
|--------------------------|--|--------------|-------------------|
| Test Case: | TC-022 | Nome: | Creazione cliente |
| Riferimento: | REQ-009 | | |
| Descrizione: | Creazione di un cliente. | | |
| Prerequisiti: | Avere effettuato il login. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Andare nella pagina Clients 2. Premere su New Client 3. Inserire i dati e premere Save. | | |
| Risultati attesi: | Il cliente si ritrova nella tabella dei clienti. | | |

| | | | |
|--------------------------|--|--------------|------------------|
| Test Case: | TC-023 | Nome: | Modifica Cliente |
| Riferimento: | REQ-009 | | |
| Descrizione: | Modificare un cliente. | | |
| Prerequisiti: | Avere effettuato il login. Trovare nella pagina Clients. Avere un cliente (TC-022). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere su Edit nella riga del cliente. 2. Modificare i dati 3. Salvare | | |
| Risultati attesi: | Il cliente appare con i nuovi valori. | | |

| | | | |
|--------------------------|--|--------------|-----------------|
| Test Case: | TC-024 | Nome: | Elimina cliente |
| Riferimento: | REQ-009 | | |
| Descrizione: | Eliminare un cliente | | |
| Prerequisiti: | Avere effettuato il login. Trovare nella pagina Clients. Avere un cliente (TC-022). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sul pulsante Delete sulla riga del cliente 2. Confermare l'eliminazione. | | |
| Risultati attesi: | Cliente eliminato (archiviato per mantenere la coerenza dei dati). | | |

| | | | |
|--------------------------|---|--------------|-----------------|
| Test Case: | TC-025 | Nome: | Grafico Cliente |
| Riferimento: | REQ-009 | | |
| Descrizione: | Visualizzare statistiche in un grafico. | | |
| Prerequisiti: | Avere effettuato il login. Avere un cliente (TC-022) e un membro del team nell'applicativo (TC-027). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Aggiungere un orologio con stato "Sold" allo storage e inserire come cliente il cliente creato prima, come data di vendita inserire quella odierna. 2. Andare in Clients. | | |
| Risultati attesi: | Sul grafico appare che questo cliente ha comprato 1 orologio. | | |

| | | | |
|--------------------------|---|--------------|------------------|
| Test Case: | TC-026 | Nome: | Dettagli Cliente |
| Riferimento: | REQ-009 | | |
| Descrizione: | Visualizzare i dettagli di un cliente. | | |
| Prerequisiti: | Avere effettuato il login. Avere un cliente (TC-022) con almeno un orologio comprato (TC-025). | | |
| Procedura: | 1. Premere sul pulsante Details nella riga del cliente. | | |
| Risultati attesi: | Si apre una schermata di informazioni sul cliente, orologi comprati, spesa totale, ecc. | | |

| | | | |
|--------------------------|--|--------------|-----------------|
| Test Case: | TC-027 | Nome: | Aggiungere Team |
| Riferimento: | REQ-008 | | |
| Descrizione: | Aggiunta di un membro del team | | |
| Prerequisiti: | Avere effettuato il login. Trovare nella pagina Team. | | |
| Procedura: | 1. Premere New Team member 2. Inserire dati validi 3. Premere Save | | |
| Risultati attesi: | Il nuovo membro del team è visibile nella lista. | | |

| | | | |
|--------------------------|--|--------------|---------------|
| Test Case: | TC-028 | Nome: | Modifica Team |
| Riferimento: | REQ-008 | | |
| Descrizione: | Modifica di un membro del team | | |
| Prerequisiti: | Avere effettuato il login. Trovare nella pagina Team e avere un membro del team (TC-027) | | |
| Procedura: | 1. Premere sul pulsante Modify sulla riga del membro aggiunto in TC-027 2. Cambiare il nome 3. Salvare | | |
| Risultati attesi: | Membro del team modificato | | |

| | | | |
|--------------------------|---|--------------|-------------------|
| Test Case: | TC-029 | Nome: | Eliminazione Team |
| Riferimento: | REQ-008 | | |
| Descrizione: | Eliminazione membro team. | | |
| Prerequisiti: | Avere effettuato il login. Trovare nella pagina Team e avere un membro del team (TC-027) | | |
| Procedura: | 1. Premere delete e confermare | | |
| Risultati attesi: | Membro del team eliminato | | |

| | | | |
|--------------------------|---|--------------|--------------|
| Test Case: | TC-030 | Nome: | Grafico Team |
| Riferimento: | REQ-009 | | |
| Descrizione: | Visualizzare statistiche in un grafico. | | |
| Prerequisiti: | Avere effettuato il login. Avere un membro del team (TC-027) e un cliente nell'applicativo (TC-022). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Aggiungere un orologio con stato "Sold" allo storage e inserire come team il membro del team creato prima (non necessario se già fatto con questo membro del team nel TC-025). 2. Andare in team. | | |
| Risultati attesi: | Sul grafico appare che questo membro del team ha venduto 1 orologio. | | |

| | | | |
|--------------------------|--|--------------|---------------|
| Test Case: | TC-031 | Nome: | Dettagli Team |
| Riferimento: | REQ-009 | | |
| Descrizione: | Visualizzare i dettagli di un membro del team. | | |
| Prerequisiti: | Avere effettuato il login. Avere un membro del team (TC-027) con almeno un orologio venduto (TC-030). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sul pulsante Details nella riga del team. | | |
| Risultati attesi: | Si apre una schermata di informazioni sul membro del team, orologi venduti, ecc. | | |

| | | | |
|--------------------------|--|--------------|--------------------|
| Test Case: | TC-032 | Nome: | Domanda DB Chatbot |
| Riferimento: | REQ-004 | | |
| Descrizione: | Domande al bot | | |
| Prerequisiti: | Avere effettuato il login. Avere un membro del team e questo deve aver venduto qualcosa. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sulla finestra di chat in basso a destra. 2. Chiedere: "What did we sold?" | | |
| Risultati attesi: | Il bot risponde con gli orologi venduti (presi dalla sezione storage "Sold"). | | |

| | | | |
|--------------------------|---|--------------|--------------------------|
| Test Case: | TC-033 | Nome: | Domanda generica Chatbot |
| Riferimento: | REQ-004 | | |
| Descrizione: | Domande al bot | | |
| Prerequisiti: | Avere effettuato il login. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sulla finestra di chat in basso a destra. 2. Chiedere: "What watches do you recommend for a student?" | | |
| Risultati attesi: | Il bot risponde con gli orologi che sono consigliati per gli studenti. | | |

| | | | |
|--------------------------|--|--------------|-------------------------|
| Test Case: | TC-033 | Nome: | Domanda Storage Chatbot |
| Riferimento: | REQ-004 | | |
| Descrizione: | Domande al bot | | |
| Prerequisiti: | Avere effettuato il login. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sulla finestra di chat in basso a destra. 2. Chiedere: "What is in my storage as unsold" | | |
| Risultati attesi: | Il bot risponde con gli orologi presenti nello storage con stato diverso da "Sold". | | |

| | | | |
|--------------------------|--|--------------|-------------------|
| Test Case: | TC-034 | Nome: | Filtri di ricerca |
| Riferimento: | REQ-004 | | |
| Descrizione: | Filtri di ricerca degli orologi. | | |
| Prerequisiti: | Avere effettuato il login e trovarsi nella home | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sulla barra di ricerca. 2. Impostare il filtro Dial color a Diamonds | | |
| Risultati attesi: | Vengono visualizzati solo gli orologi il colore del quadrante uguale a "Diamonds". | | |

| | | | |
|--------------------------|---|--------------|----------------------------|
| Test Case: | TC-035 | Nome: | Filtri di ricerca numerici |
| Riferimento: | REQ-004 | | |
| Descrizione: | Filtri di ricerca degli orologi. | | |
| Prerequisiti: | Avere effettuato il login e trovarsi nella home | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere sulla barra di ricerca. 2. Immettere come filtro diameter da 38 a 40mm. | | |
| Risultati attesi: | Vengono visualizzati solo gli orologi con il diametro da 38 a 40mm. | | |

| | | | |
|--------------------------|---|--------------|-----------------|
| Test Case: | TC-036 | Nome: | Orologio Custom |
| Riferimento: | REQ-007 | | |
| Descrizione: | Aggiunta di un orologio custom | | |
| Prerequisiti: | Avere effettuato il login e trovarsi nello storage. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere su Custom Watches 2. Premere su Add Custom Watch 3. Inserire dati validi | | |
| Risultati attesi: | Il nuovo orologio è stato aggiunto. | | |

| | | | |
|--------------------------|---|--------------|----------------------------------|
| Test Case: | TC-037 | Nome: | Aggiunta storage orologio custom |
| Riferimento: | REQ-006 | | |
| Descrizione: | Aggiunta di un orologio custom allo storage | | |
| Prerequisiti: | Avere effettuato il login e trovarsi nello storage. Avere aggiunto un orologio custom (TC-036). | | |
| Procedura: | <ol style="list-style-type: none"> 1. Premere su Custom Watches 2. Premere sull'orologio custom. 3. Premere Add to Storage 4. Inserire dati validi. | | |
| Risultati attesi: | l'orologio custom è stato aggiunto allo storage. | | |

| | | | |
|--------------------------|--|--------------|-------------------------------|
| Test Case: | TC-038 | Nome: | Centro di comunicazione email |
| Riferimento: | REQ-005 | | |
| Descrizione: | Visualizzazione delle email dell'azienda | | |
| Prerequisiti: | Avere effettuato il login e trovarsi nella home. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Andare nella pagina Communication | | |
| Risultati attesi: | Si visualizzano tutte le email dell'azienda in un punto unico. | | |

| | | | |
|--------------------------|--|--------------|---------------------------------|
| Test Case: | TC-039 | Nome: | Validazione Input registrazione |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se viene inserito un'e-mail errata, si visualizza un messaggio di errore. | | |
| Prerequisiti: | Trovare nella pagina di registrazione. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Inserire i seguenti dati: username: "Test", e-mail: "test", password: "Admin\$00" | | |
| Risultati attesi: | Si visualizza l'errore: "Registration failed, please provide an correct email!" | | |

| | | | |
|--------------------------|--|--------------|---------------------------------|
| Test Case: | TC-040 | Nome: | Validazione Input registrazione |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se viene inserito un'e-mail già usata, si visualizza un messaggio di errore. | | |
| Prerequisiti: | Trovare nella pagina di registrazione. | | |
| Procedura: | <ol style="list-style-type: none"> 1. Inserire i seguenti dati: username: "Test", e-mail: "tom@mail.com", password: "Admin\$00" | | |
| Risultati attesi: | Si visualizza l'errore: "Failed, email is already used!" | | |

| | | | |
|--------------------------|---|--------------|---------------------------------|
| Test Case: | TC-041 | Nome: | Validazione Input registrazione |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se si lascia uno o più campi vuoti, si visualizza un messaggio di errore. | | |
| Prerequisiti: | Trovarsi nella pagina di registrazione. | | |
| Procedura: | 1. Inserire i seguenti dati: username: "Test", e-mail: "", password: "" | | |
| Risultati attesi: | Si visualizza l'errore: "Registration failed, please fill all fields!" | | |

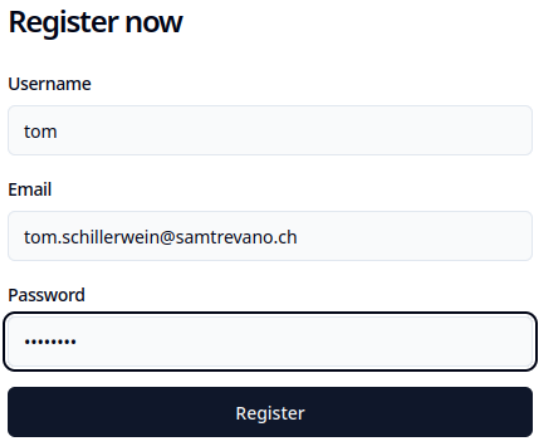
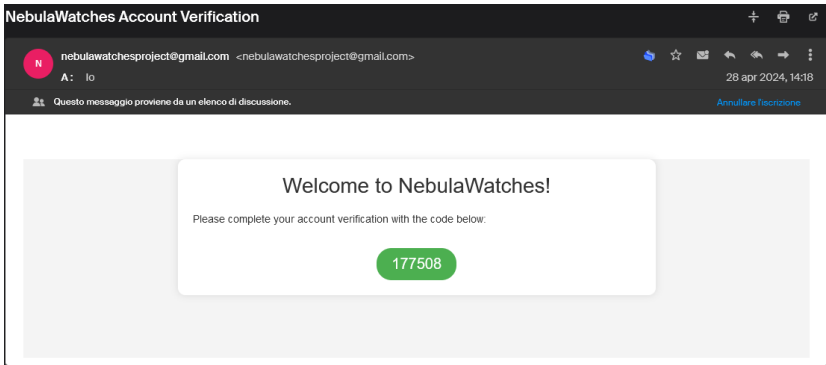
| | | | |
|--------------------------|---|--------------|---------------------------------|
| Test Case: | TC-042 | Nome: | Validazione Input registrazione |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se si inserisce una password troppo facile, si visualizza un messaggio di errore. | | |
| Prerequisiti: | Trovarsi nella pagina di registrazione. | | |
| Procedura: | 1. Inserire i seguenti dati: username: "Test", e-mail: "tescase@gmail.com", password: "password" | | |
| Risultati attesi: | Si visualizza l'errore: "Password must be at least 5 characters long and contain at least 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special character." | | |

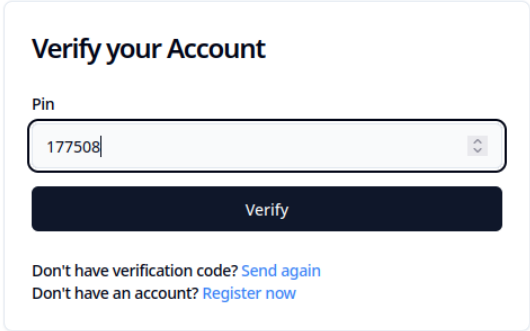
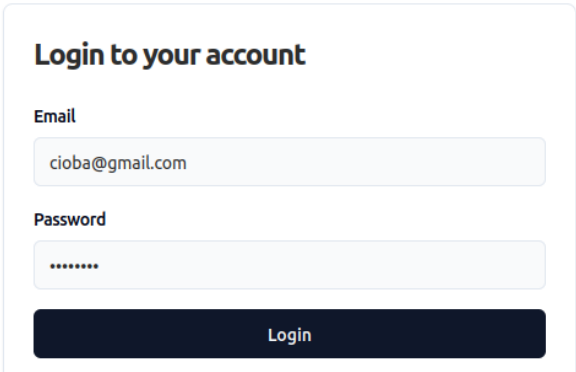

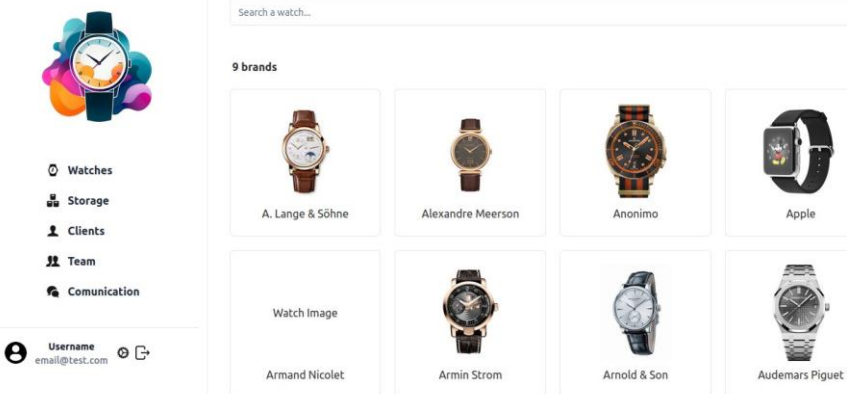
| | | | |
|--------------------------|--|--------------|---------------------------------|
| Test Case: | TC-043 | Nome: | Validazione Input registrazione |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se si inserisce una password troppo lunga, si visualizza un messaggio di errore. | | |
| Prerequisiti: | Trovarsi nella pagina di registrazione. | | |
| Procedura: | 1. Inserire i seguenti dati: username: "Test", e-mail: "tescase@gmail.com", password: "Admin\$00aaaaaaaaaaaaaaaaaaaaa" | | |
| Risultati attesi: | Si visualizza l'errore: "Failed, password too long! Max 25 characters." | | |

| | | | |
|--------------------------|---|--------------|----------------------------|
| Test Case: | TC-044 | Nome: | Validazione prezzo storage |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se si inserisce un prezzo più basso di quello di acquisto, viene richiesta la conferma. | | |
| Prerequisiti: | Avere effettuato il login. Avere un orologio nello storage con Status "Owned" nello storage. | | |
| Procedura: | 1. Premere Edit e impostare il prezzo inferiore a quello di acquisto. | | |
| Risultati attesi: | Si visualizza un messaggio di conferma. | | |

| | | | |
|--------------------------|---|--------------|--------------------------|
| Test Case: | TC-045 | Nome: | Validazione data storage |
| Riferimento: | REQ-Funzionale | | |
| Descrizione: | Se si inserisce una data di vendita inferiore a quella di acquisto, si visualizza un messaggio di errore. | | |
| Prerequisiti: | Avere effettuato il login. Avere un orologio nello storage con Status "Owned" nello storage. | | |
| Procedura: | 1. Premere Edit e impostare la data di vendita inferiore a quella di acquisto. | | |
| Risultati attesi: | Si visualizza l'errore: "The sell date must be after the purchase date!" | | |

5.2 Risultati test

| Test case | Risultato | Data | Stato |
|-----------|---|------------|----------|
| TC-001 |  <p>Premere su register e ci si trova nella pagina di verifica dell'e-mail.</p> | 03.05.2024 | Passato |
| TC-002 | <p>Mi sono registrato prima con l'e-mail tom.schillerwein@samstrevano.ch</p> <p>Nella casella postale trovo un'e-mail che arriva da nebulawatchesproject@gmail.com:</p>  <p>Inserire il pin nel form:</p> | 03.05.2024 | Parziale |

| | | | |
|--------|--|------------|---------|
| |  <p>Utente è verificato e si ritrova nel sito.</p> <p>Questo Test non funziona sul server della scuola, ma funziona da macchina locale.</p> | | |
| TC-003 |  <p>Premere Login e ci si trova nella homepage.</p> | 03.05.2024 | Passato |
| TC-004 |  <p>Premere la porta per eseguire il logout. Dopodiché ci si trova nella pagina di login</p> | 03.05.2024 | Passato |
| TC-005 | Anche se si trova il pulsante Login con Google sull'applicativo, questa feature non è implementata. | 03.05.2024 | Fallito |
| TC-006 |  <p>Premere "A. Lange & Söhne"</p> | 03.05.2024 | Passato |



Username
email@test.com

Search a watch ...

< brand A. Lange & Söhne

9 families



Premere “Lange 1”

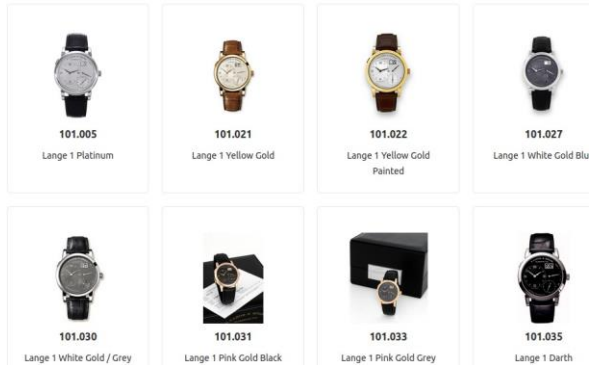


Username
email@test.com

Search a watch ...

< brand A. Lange & Söhne family Lange 1

107 watches



Premere “Lange 1 Yellow Gold”



Username
email@test.com

Search a watch ...

< brand A. Lange & Söhne family Lange 1



Reference
101.021

Name
Lange 1 Yellow Gold

Description
The ref. 101.021 Lange 1 is one of the most classic configurations of the iconic Lange 1. It f silver(ish) dial and a leather strap.

Production period
1995 - 2015

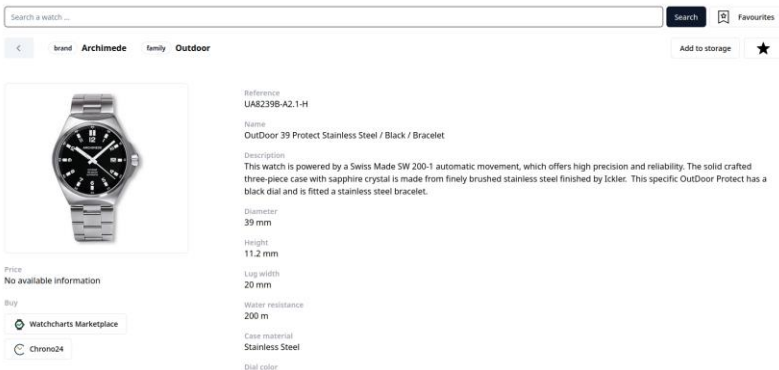
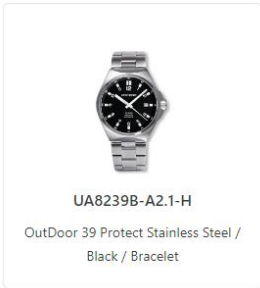
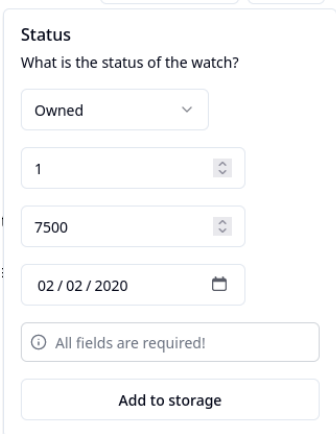
Diameter
38.5 mm


Height
9.8 mm






Case material
Yellow Gold




Dial color
Silver

Ecco tutte le informazioni di questo orologio


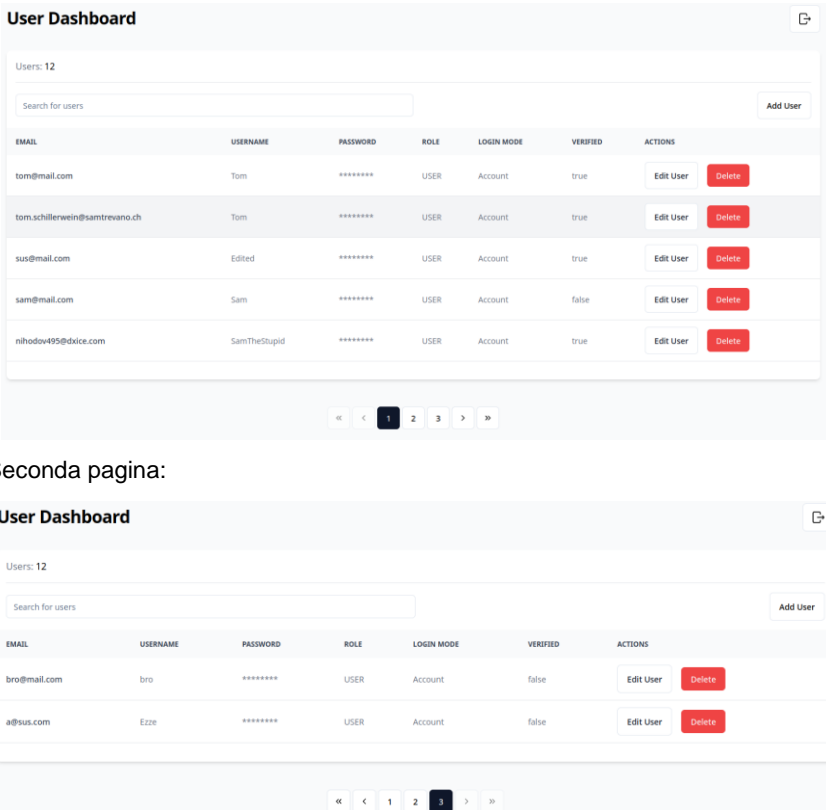
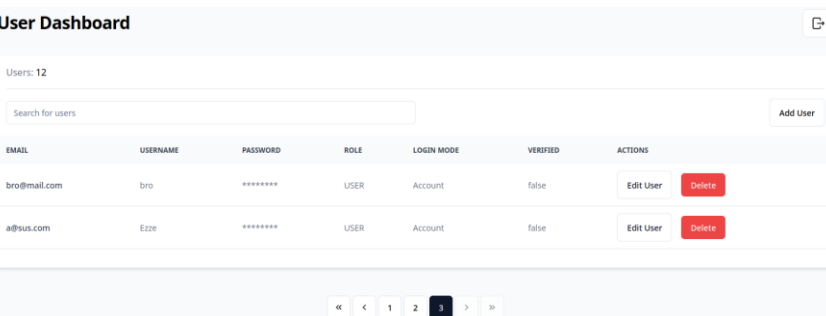
| | | | |
|--------|---|------------|---------|
| TC-007 | <p>Premere sulla stella, che diventa nera.</p>  <p>Premere su Favourites.</p>  <p>L'orologio è stato aggiunto ai preferiti.</p> | 03.05.2024 | Passato |
| TC-008 | Tornare sull'orologio di TC-007 e premere sulla stella di colore nero. Dopo che essa è diventata bianca premere su Favourites e si può notare l'orologio non è più presente. | 03.05.2024 | Passato |
| TC-009 |  <p>Dopo aver premuto su Add to storage, andare in storage. Se si preme sull'orologio si vedono le informazioni.</p> | 03.05.2024 | Passato |

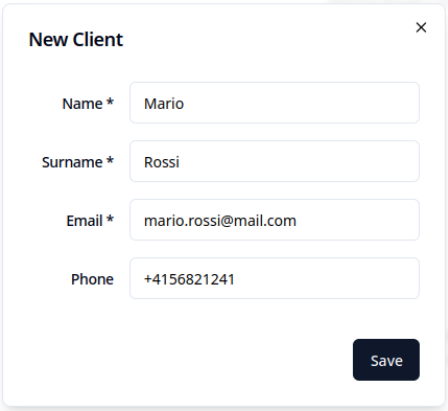
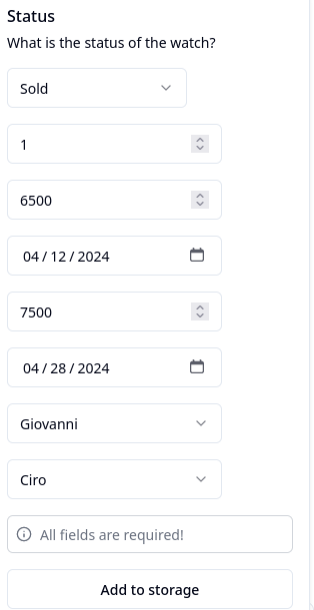
| | | | |
|--------|---|------------|---------|
| | <div>Storage</div> <div>Owned Sold</div> <div>  <p>1 pcs.</p> <p>706.025</p> <p>Tourbograph Perpetual 'Pour le Mérite' Platinum / Silver</p> </div> <div> <p>Informations</p> <p>Reference 706.025</p> <p>Name Tourbograph Perpetual 'Pour le Mérite' Platinum / Silver</p> <p>Amount 1</p> <p>Status Owned</p> <p>Bought for 7500 Fr.</p> <p>Purchase date 02/02/2020</p> </div> | | |
| TC-010 | <div> <p>Status</p> <p>What is the status of the watch?</p> <p>Sold</p> <p>1</p> <p>7500</p> <p>01 / 01 / 2022</p> <p>80000</p> <p>05 / 05 / 2023</p> <p>John</p> <p>Arthur</p> <p>All fields are required!</p> <p>Add to storage</p> </div> <p>Dopo aver aggiunto allo storage l'orologio già venduto, si può visualizzare nella pagina storage.</p> | 03.05.2024 | Passato |

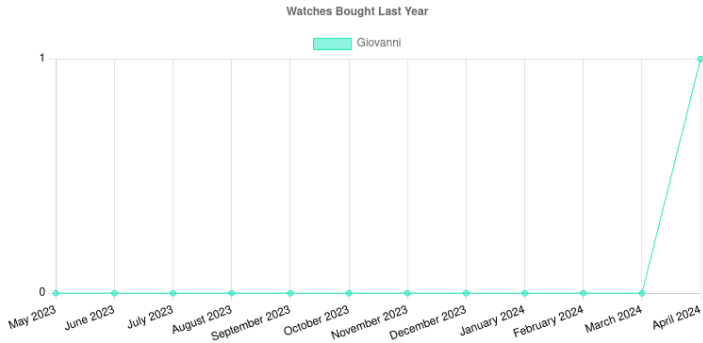
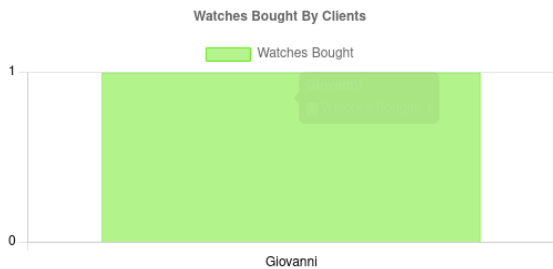


| | | | |
|--------|--|------------|---------|
| | <div> <div>Storage</div> <div> <div>Owned</div> <div>Sold</div> </div> <div> <div>1 pcs.</div> <div>  </div> <div>AM-1002.06.004.A06</div> <div>Nautilo Automatic Stainless Steel / Blue / Leather</div> </div> <div> <div>Informations</div> <div> <div>Reference</div> <div>AM-1002.06.004.A06</div> </div> <div> <div>Name</div> <div>Nautilo Automatic Stainless Steel / Blue / Leather</div> </div> <div> <div>Amount</div> <div>1</div> </div> <div> <div>Status</div> <div>Sold</div> </div> <div> <div>Bought for</div> <div>7500 Fr.</div> </div> <div> <div>Purchase date</div> <div>01/01/2022</div> </div> <div> <div>Sold for</div> <div>80000 Fr.</div> </div> <div> <div>Sale date</div> <div>05/05/2023</div> </div> <div> <div>Client</div> <div>John Marston</div> </div> <div> <div>Team</div> <div>Arthur Morgan</div> </div> </div> </div> | | |
| TC-011 | <div> <div>Storage</div> <div> <div>Owned</div> <div>Sold</div> </div> <div> <div>1 pcs.</div> <div>  </div> <div>706.025</div> <div>Tourbograph Perpetual 'Pour le Mérite' Platinum / Silver</div> </div> <p>Dopo aver premuto la conferma di eliminazione, l'orologio non è più presente nello storage.</p> </div> | 03.05.2024 | Passato |
| TC-012 | <div> <div>Paginazione:</div> <div> <div>  <div>Blancpain</div> </div> <div>  <div>Breguet</div> </div> <div>  <div>Breitling</div> </div> <div> <div>«</div> <div><</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>...</div> <div>8</div> <div>></div> <div>»</div> </div> </div> </div> | 03.05.2024 | Passato |

| | | | |
|--------|--|------------|---------|
| TC-013 | <p>Dopo la ricerca viene visualizzato il risultato.</p> <div> <p>1 watches</p> <div> <p>Limited to 1 unit</p>  <p>140.049</p> <p>Zeitwerk Duncan Wang / Kidz Horizon</p> </div> </div> | 03.05.2024 | Passato |
| TC-014 | <div> <p>Set Watch Sold: ×</p> <p>Here you can edit this watch when you sell it:</p> <ul style="list-style-type: none"> • Current amount: 1 • Current status: Owned • Bought for: 7500 • Purchase date: 02/02/2020 <div> <p>Amount <input type="text" value="1"/></p> <p>Price <input type="text" value="9000"/></p> <p>Sell Date <input type="text" value="04 / 04 / 2021"/></p> <p>Client <input type="text" value="John"/></p> <p>Team <input type="text" value="Arthur"/></p> <p><input type="button" value="Save Changes"/></p> <p><input type="button" value="All fields are required!"/></p> </div> <p>Nella tab "Sold" dello storage si trova l'orologio appena modificato.</p> <div> <p>Storage</p> <p><input type="button" value="Owned"/> <input type="button" value="Sold"/></p> <div> <p>1 pcs. </p>  <p>706.025</p> <p>Tourbograph Perpetual 'Pour le Mérite' Platinum / Silver</p> </div> </div> </div> | 03.05.2024 | Passato |


| | | | | |
|--------|---|--|------------|---------|
| TC-015 | Prezzo: |  | 03.05.2024 | Passato |
| TC-016 | Fare il login con un utente admin. |  | 03.05.2024 | Passato |
| TC-017 | Premere sul pulsante Add User |  <p>Utente aggiunto:</p> <div> testcase@mail.com Test Case </div> | 03.05.2024 | Passato |
| TC-018 | Cambio Username da "Test Case" a "Modified" | <div> testcase@mail.com Modified </div> | 03.05.2024 | Passato |




| | | | |
|--------|--|------------|---------|
| TC-019 | <p>Dopo aver premuto Delete, l'utente non è più presente.</p> <p>Prima:</p> <p>Users: 11</p> <p>Dopo:</p> <p>Users: 10</p> | 03.05.2024 | Passato |
| TC-020 | <p>Ricerca "tom" (questi utenti sono stati inseriti nell'applicativo):</p>  | 03.05.2024 | Passato |
| TC-021 | <p>Multiple pagine di utenti:</p>  <p>Seconda pagina:</p>  | 03.05.2024 | Passato |

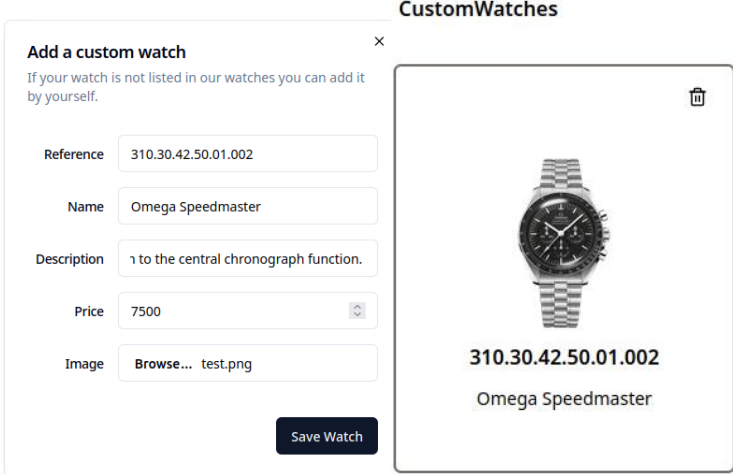
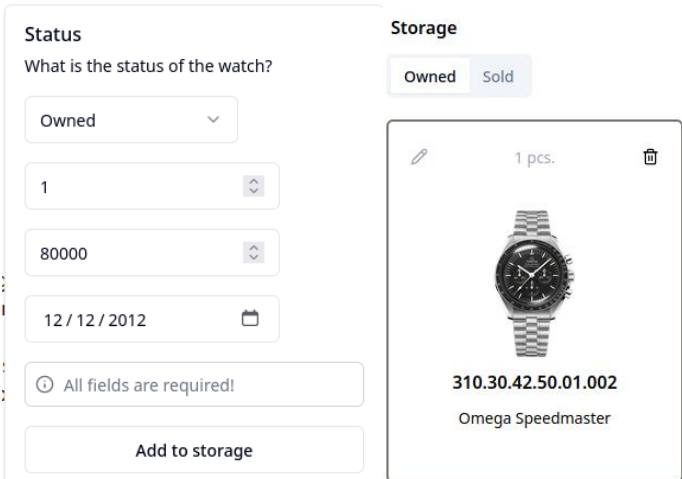
| TC-022 | <p>Inserisco un nuovo cliente:</p>  <p>Risultato:</p> <table> <thead> <tr> <th>Name</th><th>Surname</th><th>Email</th><th>Phone</th></tr> </thead> <tbody> <tr> <td>Mario</td><td>Rossi</td><td>mario.rossi@mail.com</td><td>+4156821241</td></tr> </tbody> </table> | Name | Surname | Email | Phone | Mario | Rossi | mario.rossi@mail.com | +4156821241 | 03.05.2024 | Passato |
|----------|---|----------------------|-------------|-------|-------|----------|-------|----------------------|-------------|------------|---------|
| Name | Surname | Email | Phone | | | | | | | | |
| Mario | Rossi | mario.rossi@mail.com | +4156821241 | | | | | | | | |
| TC-023 | <p>Modifica cliente, cambio il nome da Mario in Giovanni:</p> <table> <thead> <tr> <th>Name</th><th>Surname</th><th>Email</th><th>Phone</th></tr> </thead> <tbody> <tr> <td>Giovanni</td><td>Rossi</td><td>mario.rossi@mail.com</td><td>+4156821241</td></tr> </tbody> </table> | Name | Surname | Email | Phone | Giovanni | Rossi | mario.rossi@mail.com | +4156821241 | 03.05.2024 | Passato |
| Name | Surname | Email | Phone | | | | | | | | |
| Giovanni | Rossi | mario.rossi@mail.com | +4156821241 | | | | | | | | |
| TC-024 | Utente viene eliminato. | 03.05.2024 | Passato | | | | | | | | |
| TC-025 | <p>Dati dell'orologio venduto:</p>  | 03.05.2024 | Passato | | | | | | | | |

| | | | |
|--------|---|------------|---------|
| | <p>Grafico del cliente 1:</p>  <p>Grafico del cliente 2:</p>  | | |
| TC-026 | <p>Informazioni sul cliente:</p> <p>Informations of Giovanni Rossi Email: mario.rossi@mail.com Phone: +4156821241</p> <p>Total watches buys: 1</p> <p>Total watches expenses: 7500 CHF</p> <div><p>1 pcs. </p><p>AM-4000.04.441.W88 Epurato Automatic Bronze / Anthracite / Leather</p></div> | 03.05.2024 | Passato |

| TC-027 | <div><div><div>New Team Member</div><div><div><div>Name *</div><div>Luigi</div></div><div><div>Surname *</div><div>Romano</div></div><div><div>Email *</div><div>luigi.romano@mail.com</div></div><div><div>Phone</div><div>+4156155016</div></div><div><div>Role</div><div>Scout</div></div></div><div><div>Save</div></div></div><div>Nuovo membro del team aggiunto:</div><table><thead><tr><th>Name</th><th>Surname</th><th>Email</th></tr></thead><tbody><tr><td>Luigi</td><td>Romano</td><td>luigi.romano@mail.com</td></tr></tbody></table></div> | Name | Surname | Email | Luigi | Romano | luigi.romano@mail.com | 03.05.2024 | Passato | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------|---|-----------------------|---------|----------|-------|-----------|-----------------------|------------|---------|-------------|---|----------------|---|--------------|---|---------------|---|---------------|---|--------------|---|---------------|---|------------|---|------------|---|-------|-------|----------|---|-----------|---|-----------|---|-------------|---|----------------|---|--------------|---|---------------|---|---------------|---|--------------|---|---------------|---|------------|---|------------|---|------------|---------|
| Name | Surname | Email | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Luigi | Romano | luigi.romano@mail.com | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TC-028 | <div>Modifica team, cambio il nome da Luigi in Ciro:</div> <table><thead><tr><th>Name</th><th>Surname</th><th>Email</th></tr></thead><tbody><tr><td>Ciro</td><td>Romano</td><td>luigi.romano@mail.com</td></tr></tbody></table> | Name | Surname | Email | Ciro | Romano | luigi.romano@mail.com | 03.05.2024 | Passato | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Name | Surname | Email | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Ciro | Romano | luigi.romano@mail.com | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TC-029 | Membro del team viene eliminato. | 03.05.2024 | Passato | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TC-030 | <div>Grafico 1:</div> <div><div>Watches Sold last Year</div><div><div>Ciro</div></div><table><thead><tr><th>Month</th><th>Sales</th></tr></thead><tbody><tr><td>May 2023</td><td>0</td></tr><tr><td>June 2023</td><td>0</td></tr><tr><td>July 2023</td><td>0</td></tr><tr><td>August 2023</td><td>0</td></tr><tr><td>September 2023</td><td>0</td></tr><tr><td>October 2023</td><td>0</td></tr><tr><td>November 2023</td><td>0</td></tr><tr><td>December 2023</td><td>0</td></tr><tr><td>January 2024</td><td>0</td></tr><tr><td>February 2024</td><td>0</td></tr><tr><td>March 2024</td><td>0</td></tr><tr><td>April 2024</td><td>1</td></tr></tbody></table></div> <div>Grafico 2:</div> <div><div>Sold Watches by Team</div><div><div>Watches Sold</div></div><table><thead><tr><th>Month</th><th>Sales</th></tr></thead><tbody><tr><td>May 2023</td><td>0</td></tr><tr><td>June 2023</td><td>0</td></tr><tr><td>July 2023</td><td>0</td></tr><tr><td>August 2023</td><td>0</td></tr><tr><td>September 2023</td><td>0</td></tr><tr><td>October 2023</td><td>0</td></tr><tr><td>November 2023</td><td>0</td></tr><tr><td>December 2023</td><td>0</td></tr><tr><td>January 2024</td><td>0</td></tr><tr><td>February 2024</td><td>0</td></tr><tr><td>March 2024</td><td>0</td></tr><tr><td>April 2024</td><td>1</td></tr></tbody></table></div> | Month | Sales | May 2023 | 0 | June 2023 | 0 | July 2023 | 0 | August 2023 | 0 | September 2023 | 0 | October 2023 | 0 | November 2023 | 0 | December 2023 | 0 | January 2024 | 0 | February 2024 | 0 | March 2024 | 0 | April 2024 | 1 | Month | Sales | May 2023 | 0 | June 2023 | 0 | July 2023 | 0 | August 2023 | 0 | September 2023 | 0 | October 2023 | 0 | November 2023 | 0 | December 2023 | 0 | January 2024 | 0 | February 2024 | 0 | March 2024 | 0 | April 2024 | 1 | 03.05.2024 | Passato |
| Month | Sales | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| May 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| June 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| July 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| August 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| September 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| October 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| November 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| December 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| January 2024 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| February 2024 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| March 2024 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| April 2024 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Month | Sales | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| May 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| June 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| July 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| August 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| September 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| October 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| November 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| December 2023 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| January 2024 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| February 2024 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| March 2024 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| April 2024 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|--------|--|------------|---------|
| TC-031 | <p>Informazioni di Ciro Romano:</p> <p>Informations of Ciro Romano Email: luigi.romano@mail.com Phone: +4156155016 Role: Scout Total watches sold: 1</p> <div> <p>1 pcs.</p>  <p>AM-4000.04.441.W88 Epurato Automatic Bronze / Anthracite / Leather</p> </div> | 03.05.2024 | Passato |
| TC-032 | <p>Risposta del bot:</p> <p>what did we sold?</p> <p>The sold watches are:</p> <ol style="list-style-type: none"> 1. Reference: 234.032 (1815 Up/Down Pink Gold) 2. Reference: 26521PT.YY.1220PT.01 (Royal Oak Ultra Thin Tourbillon Platinum / Smoked Blue Evolutive / Sapphire) | 03.05.2024 | Passato |
| TC-033 | <p>Risposta del bot:</p> <p>what watches do you recommend for a student</p> <ol style="list-style-type: none"> 1. Casio G-Shock: A durable and affordable option that is perfect for everyday wear and outdoor activities. 2. Timex Weekender: A classic and stylish watch that is versatile enough to be worn with casual or dressy outfits. 3. Seiko 5 Sports: A reliable and well-made automatic watch that offers great value for its price. | 03.05.2024 | Passato |
| TC-033 | <p>Risposta del bot:</p> <p>what is in my storage as unsold</p> <p>The watch '79030B-0001' (also known as M79030B-0001, 79030B0001, BB58, Black Bay 58) is currently in your storage as unsold. It was purchased on May 2, 2024, and is still owned by you.</p> | 03.05.2024 | Passato |

| | | | |
|--------|---|------------|---------|
| TC-034 | <p>Filtro:</p> <p>Dial colors (23)</p> <p><input type="checkbox"/> Champagne <input checked="" type="checkbox"/> Diamonds</p> <p>Risultato:</p> <p>2 watches</p> <div>  <p>26242OR.ZZ.1322OR.02 Royal Oak Chronograph 41 Pink Gold / Diamond</p> </div> <div>  <p>26242OR.ZZ.1322OR.01 Royal Oak Chronograph 41 Pink Gold / Diamond / 50th</p> </div> | 03.05.2024 | Passato |
| TC-035 | <p>Filtro:</p> <p>Min. Max.</p> <p>38 40</p> <p>Vengono solo visualizzati orologi con diametro tra 38 e 40 mm.</p> <div>  <p>Diameter 39 mm</p> <p>Height 8.9 mm</p> <p>Lug width 20 mm</p> <p>Case material Silver</p> <p>Dial color Blue</p> <p>Glass material</p> </div> | 03.05.2024 | Passato |

| | | | |
|--------|---|------------|---------|
| TC-036 | <p>Orologio custom aggiunto:</p>  | 03.05.2024 | Passato |
| TC-037 | <p>Inserimento dati e visualizzazione nello storage:</p>  | 03.05.2024 | Passato |
| TC-038 | <p>Non si visualizzano le e-mail, ma questo messaggio:</p> <div data-bbox="454 1435 962 1619"> <p>Work in progress!</p> <p>Unfortunately this feature isn't ready yet! Stay tuned for more.</p> </div> | 03.05.2024 | Fallito |

| | | | |
|--------|---|------------|---------|
| TC-039 | <p>Risultato:</p> <p>Register now</p> <p>Username</p> <input type="text" value="Test"/> <p>Email</p> <input type="text" value="test"/> <p>Password</p> <input type="password" value="*****"/> <p>Register</p> <p>Error Registration failed, please provide an correct email!</p> | 03.05.2024 | Passato |
| TC-040 | <p>Risultato:</p> <p>Register now</p> <p>Username</p> <input type="text" value="Test"/> <p>Email</p> <input type="text" value="tom@mail.com"/> <p>Password</p> <input type="password" value="*****"/> <p>Register</p> <p>Error Failed, email is already used!</p> | 03.05.2024 | Passato |
| TC-041 | <p>Risultato:</p> <p>Register now</p> <p>Username</p> <input type="text" value="Test"/> <p>Email</p> <input type="text" value="example@gmail.com"/> <p>Password</p> <input type="text" value="YourPassword"/> <p>Register</p> <p>Error Registration failed, please fill all fields!</p> | 03.05.2024 | Passato |

| | | | |
|--------|--|------------|---------|
| TC-042 | <p>Risultato:</p> <p>Register now</p> <p>Username</p> <input type="text" value="Test"/> <p>Email</p> <input type="text" value="testcase@gmail.com"/> <p>Password</p> <input type="password" value="*****"/> <p>Register</p> <p>Error Password must be at least 5 characters long and contain at least 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special character.</p> | 03.05.2024 | Passato |
| TC-043 | <p>Risultato:</p> <p>Register now</p> <p>Username</p> <input type="text" value="Test"/> <p>Email</p> <input type="text" value="testcase@gmail.com"/> <p>Password</p> <input type="password" value="*****"/> <p>Register</p> <p>Error Failed, password too long! Max 25 characters</p> | 03.05.2024 | Passato |

| | | | |
|--------|---|------------|---------|
| TC-044 | <p>Prezzo di vendita inferiore a quello di acquisto (6000 < 7500):</p> <p>Set Watch Sold: ×</p> <p>Here you can edit this watch when you sell it:</p> <ul style="list-style-type: none"> • Current amount: 1 • Current status: Owned • Bought for: 7500 • Purchase date: 01/01/2022 <p>Amount <input type="text" value="1"/></p> <p>Price <input type="text" value="6000"/></p> <p>Sell Date <input type="text" value="01 / 01 / 2022"/></p> <p>Client <input type="text" value="test"/></p> <p>Team <input type="text" value="Gio"/></p> <p>Messaggio di conferma:</p> <p>Attention!</p> <p>The sell price is lower than the buy price! Is this correct?</p> <p><input type="button" value="Cancel"/> <input type="button" value="Yes"/></p> | 03.05.2024 | Passato |
| TC-045 | <p>Data di vendita prima della data di acquisto (impossibile):</p> <p>Set Watch Sold: ×</p> <p>Here you can edit this watch when you sell it:</p> <ul style="list-style-type: none"> • Current amount: 1 • Current status: Owned • Bought for: 150000 • Purchase date: 01/01/2022 <p>Amount <input type="text" value="1"/></p> <p>Price <input type="text" value="2000000"/></p> <p>Sell Date <input type="text" value="01 / 01 / 2021"/></p> <p>Client <input type="text" value="test"/></p> <p>Team <input type="text" value="Test"/></p> <p>Risultato:</p> <p>❗ Error The sell date must be after the purchase date!</p> | 03.05.2024 | Passato |

5.3 Mancanze/limitazioni conosciute

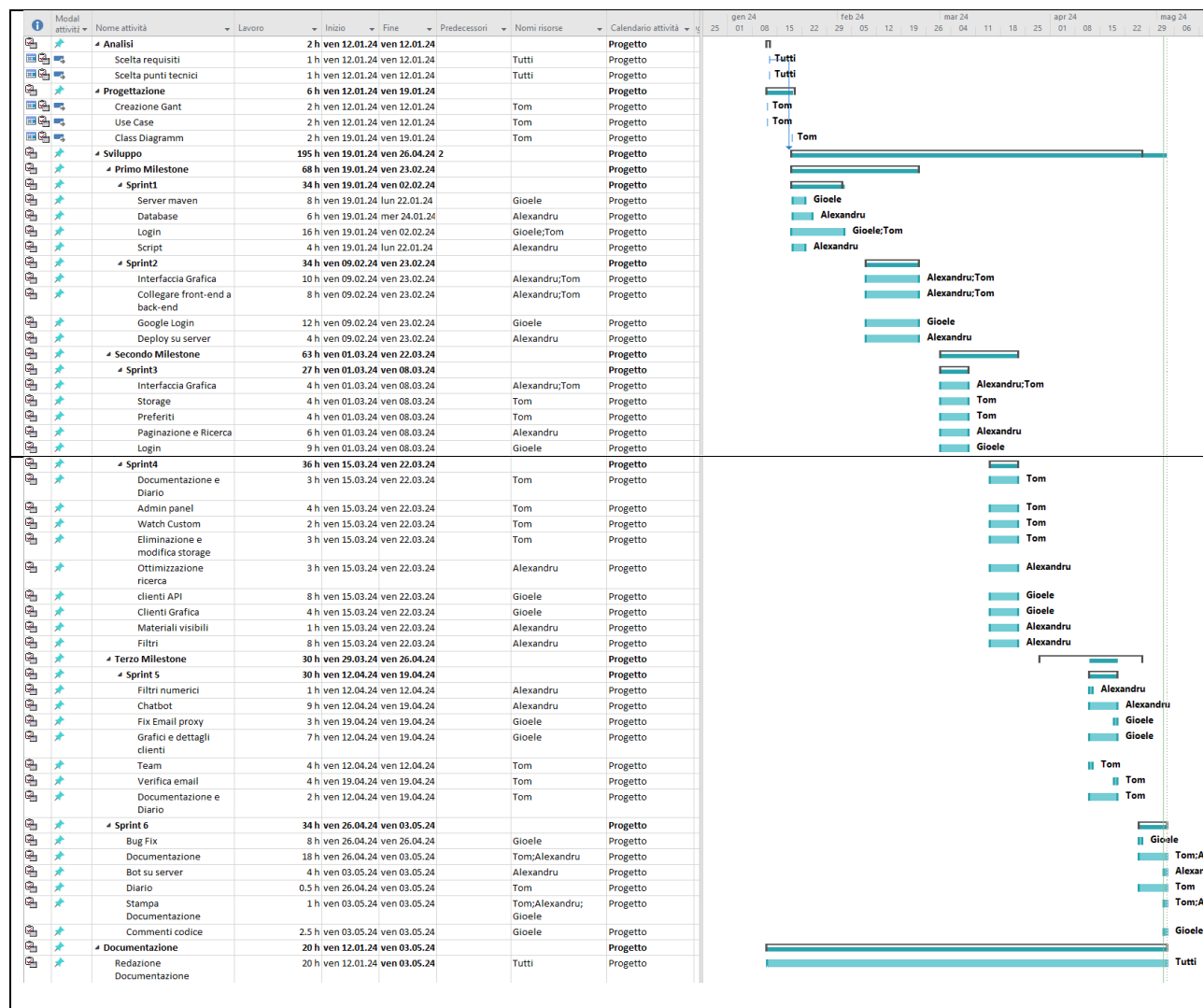
Non siamo riusciti a implementare 2 feature richieste, il centro e-mail e il login con Google.

Per il centro e-mail, non abbiamo implementato questa feature perché ci è finito il tempo e sarebbe stato una parte complicata e non molto veloce da fare.

Per il login con Google, abbiamo speso molto tempo cercando di realizzare questo, ma non siamo riusciti a far funzionare il resto del sito con il login di Google. Tuttavia, siamo riusciti a farlo funzionare in un progetto a parte.

6 Consuntivo

L'analisi e la progettazione sono stati eseguiti come programmato, ma lo sviluppo ha subito alcune modifiche. Le date degli sprint sono state modificate e perciò si è spostato tutta l'implementazione. Inoltre Abbiamo speso più tempo implementando login e altre feature, di conseguenza non è stato possibile implementare il centro di email e messaggi che inizialmente era pianificato fare nello sprint 5 e 6.



7 Conclusioni

Questo progetto è abbastanza unico a suo modo dato che non ci sono noti altri siti web per la gestione del proprio negozio di orologi. Questo sito è molto specifico, infatti il suo target è un mercato di nicchia. Comunque, con una interfaccia grafica avvincente e moderna pensiamo che questo progetto possa essere utile. Inoltre l'implementazione di tecnologie nuove come un chatbot basato sui dati del database è una feature molto interessante anche per il futuro.

7.1 Sviluppi futuri

Implementazione del centro di e-mail, implementazione del login con Google, implementare che un membro del team aggiunto nella sezione "Team" può accedere e inserire vendite all'interno dell'applicativo.

7.2 Considerazioni personali

Siamo generalmente soddisfatti de nostro progetto, anche se non completamente finito, ha molte feature utili per una gestione di un negozio.

Abbiamo imparato a lavorare in un gruppo e con una modalità agile su un progetto di un semestre. Inoltre, abbiamo imparato molto sullo sviluppo web, specialmente usando due linguaggi che a scuola non abbiamo trattato. Soprattutto lo sviluppo di un chatbot ci ha permesso di imparare molti concetti e skill utili per il futuro.

8 Glossario

| Termine | Descrizione |
|--------------|--|
| API | Application Programming Interface: insieme di strumenti e procedure che consentono a diversi software di comunicare tra loro. |
| Axios | Axios è una libreria JavaScript che consente di effettuare richieste HTTP. È ampiamente utilizzato per eseguire richieste come GET, POST, PUT, DELETE e altre operazioni CRUD verso un API. |
| CORS | Cross-Origin Resource Sharing, si tratta di una politica di sicurezza implementata nei browser web che controlla le richieste di risorse fatte da script presenti in una pagina web a un altro dominio. |
| DML | Data Manipulation Language, sottoinsieme di linguaggi di programmazione utilizzati per manipolare dati all'interno di database. Le istruzioni DML consentono di inserire, modificare, eliminare e recuperare dati da tabelle di database. |
| JWT | JSON Web Token: standard web per lo scambio di dati definito dalla RFC 7519 proposto nel 2015. |
| LLM | Large Language Model, si riferisce a modelli di intelligenza artificiale avanzati che sono addestrati su enormi quantità di testo per comprendere e generare linguaggio naturale. |
| MailJet | Mailjet è una piattaforma di email che permette di mandare email tramite Rest API. |
| OpenAI | OpenAI: azienda di intelligenza artificiale, autori di ChatGPT. |
| RAG | Retriever Reader Generator, framework utilizzato per la generazione di linguaggio naturale, in particolare nell'ambito dei modelli di linguaggio |
| Rest API | Representational State Transfer Application Programming Interface: fornisce un'interfaccia basata su HTTP per consentire ai client di accedere e manipolare le risorse del server in modo efficiente e scalabile |
| Shadcn Vue | Shadcn Vue: libreria di componenti grafici che utilizza vue e Tailwind per creare componenti grafici accattivanti. |
| Tailwind CSS | Tailwind CSS è un framework CSS open source usato per applicare velocemente uno stile alle pagine HTML. |
| Vue JS | Vue.js è un framework JavaScript open-source per la costruzione di interfacce utente e applicazioni web |

9 Bibliografia

9.1 Sitografia

- <https://spring.io/>, *varie pagine*, 26-01-2024 / 03-05-2024
- <https://chat.openai.com/>, *varie chat*, 19-01-2024 / 03-05-2024
- <https://python.langchain.com/docs>, *varie pagine*, 22-03-2024 / 03-05-2024
- <https://www.mailjet.com/>, *varie pagine*, 26-04-2024
- <https://www.shadcn-vue.com/>, *varie pagine*, 19-01-2024 / 03-05-2024
- <https://flowbite.com/>, *varie pagine*, 26-01-2024 / 16-02-2024
- <https://www.youtube.com/>, *varie pagine*, 19-01-2024 / 09-02-2024
- <https://dev.mysql.com/doc/>, *diverse pagine*, 19-01-2024 / 09-02-2024

10 Allegati

- Diari di lavoro
- Applicativo web
- Quaderno dei Compiti
- Abstract
- Gantt (preventivo e consuntivo)
- Use Case
- Diagramma ER
- Dipendenze
- Documentazione API

11 Indice Delle Figure

| | |
|---------------------------------------|----|
| 1 Use Case | 7 |
| 2 Diagramma di Gantt Preventivo | 8 |
| 3 Architettura sistema | 9 |
| 4 Architettura Chatbot | 9 |
| 5 Architettura Agent | 10 |
| 6 Schema ER | 11 |
| 7 Design della Home | 12 |
| 8 Diagramma Classi | 13 |
| 9 Struttura Frontend | 17 |
| 10 Interfaccia Home | 20 |
| 11 Struttura file progetto | 20 |