

Valutazione Discromatopsia web

Titolo del progetto: Valutazione Discromatopsia web
Alunno/a: Luca Lorenzon
Classe: I4AA
Anno scolastico: 2023/2024
Docente responsabile: Geo Petrini

1	Introduzione.....	4
1.1	Informazioni sul progetto.....	4
1.2	Abstract	4
1.3	Scopo	4
2	Analisi	5
2.1	Analisi del dominio	5
2.2	Analisi e specifica dei requisiti	5
2.2.1	Spiegazione elementi tabella dei requisiti:.....	7
2.3	Use case	8
2.3.1	Utente.....	8
2.3.2	Server.....	8
2.4	Pianificazione	9
2.5	Analisi dei mezzi.....	10
2.5.1	Software	10
2.5.2	Hardware.....	10
3	Progettazione	11
3.1	Design dell'architettura del sistema	11
3.2	Design dei dati e database.....	12
3.2.1	User.....	12
3.2.2	Album	12
3.2.3	Image	12
3.3	Design delle interfacce	13
3.3.1	Pagina login	13
3.3.2	Pagina registrazione	14
3.3.3	Pagina generazione immagini.....	15
3.3.4	Pagina history	17
3.4	Design procedurale	18
3.4.1	Diagrammi di flusso	18
4	Implementazione	21
4.1	Struttura cartelle	21
4.2	Backend	22
4.2.1	Database.....	22
4.2.2	REST API.....	25
4.2.3	Scheduled Task	29
4.3	Frontend	30
4.3.1	Pagine e navigazione.....	30
4.3.2	Comunicazione con backend.....	36
4.3.3	Multilingua	37
4.3.4	Responsive	38
4.4	Docker	39
4.4.1	Docker backend	39
4.4.2	Docker gui_web	39
4.4.3	Docker db.....	40
5	Test.....	41
5.1	Protocollo di test.....	41
5.2	Risultati test.....	45
5.3	Test REST API	46
5.4	Mancanze/limitazioni conosciute.....	47
6	Consuntivo.....	48
6.1	Differenze con Gantt preventivo.....	49
7	Conclusioni	50
7.1	Sviluppi futuri.....	50
7.2	Considerazioni personali.....	50
8	Glossario	51
9	Sitografia	52
10	Allegati	52

Figura 1 Use Case	8
Figura 2 Gantt preventivo	9
Figura 3 Schema di rete	11
Figura 4 Diagramma ER	12
Figura 5 Schema logico	12
Figura 6 Mockup pagina login	13
Figura 7 Mockup pagina registrazione	14
Figura 8 Mockup pagina principale	15
Figura 9 Mockup immagini mostrato dopo esser state generate	16
Figura 10 Mockup pagina history	17
Figura 11 Swimlane generazione immagini	18
Figura 12 Swimlane registrazione utente	19
Figura 13 Swimlane visualizzazione history	20
Figura 14 Struttura cartelle	21
Figura 15 Swimlane gestione errori	26
Figura 16 Pagina SignUp	31
Figura 17 Notifica errore username	31
Figura 18 Header Homepage	32
Figura 19 Homepage	32
Figura 20 Immagini generate mostrate a schermo	33
Figura 21 Messaggio al posto dell'history senza accesso	33
Figura 22 Messaggio se non si ha ancora generato nulla	33
Figura 23 Valore history	33
Figura 24 Contenuto valore nell'history	34
Figura 26 Modal informazioni	35
Figura 25 Pagina d'errore	35
Figura 27 Schema gestione errori	37
Figura 28 Test automatizzati	46
Figura 29 Risultato test Postman	47
Figura 30 Gantt consuntivo	48

1 Introduzione

1.1 Informazioni sul progetto

Informazioni generali sul progetto:

- Sezione: Informatica
- Nome progetto: Valutatore discromatopsia web
- Allievo: Luca Lorenzon
- Classe I4AA
- Docente responsabile: Geo Petrini
- Data inizio: 30.08.2023
- Data consegna: 07.12.2023
- Link repository: http://gitsam.cpt.local/2023_2024_1_semestre/valutatore_discromatopsia_web

1.2 Abstract

The goal of this project is to create a dyschromatopsia test available for anyone without download anything. To achieve this goal, I will create a website that uses an algorithm (made in python by Jonas Bertossa) to elaborate an image with the dyschromatopsia types with the possibility, if you log in, of saving your elaboration in a personal history. The values in the history get automatically removed after 30 days but it is possible to remove them manually. The website will be responsive therefore it will also work on mobile devices. In this document it is described the development of the whole project.

1.3 Scopo

Lo scopo del progetto è quello di creare un sito web dove sarà possibile fare dei test sulla discromatopsia con delle proprie immagini personalizzate e di poter salvare queste immagini elaborate che si utilizzano per il test nell'history in modo che le si possano guardare di nuovo.

In più questo progetto mi permetterà di migliorare le mie conoscenze di python, di react e ad imparare ad utilizzare Docker. Oltre a migliorare le mie conoscenze d'informatica, questo progetto mi permetterà anche di migliorare la mia gestione nel tempo dei progetti, essendo il primo progetto individuale che svolgiamo.

2 Analisi

2.1 Analisi del dominio

Per sviluppare questo progetto partirò dal progetto “Valutatore discromatopsia” sviluppato da Jonas Bertossa nel 2021. Questo progetto consiste in un’applicazione in python che elabora delle immagini applicando i vari livelli di discromatopsia. Partendo da questo progetto dovrò andare a creare un’interfaccia web per poter interagire con il programma in modo da non dover far installare nulla sulla macchina dell’utente per poter fare un test per la discromatopsia. Sul sito sarà possibile caricare le proprie immagini che verranno elaborate con i vari tipi di discromatopsia.

Questo prodotto potrà venir utilizzato da chiunque voglia fare un test per la discromatopsia e per utilizzare questo prodotto non bisognerà avere alcuna conoscenza teorica.

Attualmente esistono già dei test online per la discromatopsia (per esempio <https://www.it.colorlitelens.com/ishihara-test-di-daltonismo.html>) ma quello che si vuole aggiungere con questo progetto è che si possono utilizzare le proprie immagini per eseguire il test e la presenza dell’history dove verranno salvate le immagini generate in precedenza per fare il test.

2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Pagina per accesso o registrazione
Priorità	1
Versione	1.0
Note	Creare pagina per poter accedere / registrarsi al sito web con un username ed una password
Sotto requisiti	
001	Deve essere presente un tasto per fare logout dall’account nell’homepage.

ID: REQ-02	
Nome	Pagina elaborazione immagini
Priorità	1
Versione	1.0
Note	Pagina dove si potranno elaborare le immagini tramite l’applicazione “Valutatore discromatopsia”

ID: REQ-03	
Nome	Pagina history
Priorità	1
Versione	1.0
Note	Pagina dove si potranno vedere le ultime elaborazioni

ID: REQ-04	
Nome	Eliminazione valori history
Priorità	1
Versione	1.0
Note	Ci deve essere la possibilità di eliminare i valori nell'history

ID: REQ-05	
Nome	Eliminazione automatica
Priorità	1
Versione	1.0
Note	Dopo 30 giorni le elaborazioni nell'history verranno eliminate automaticamente

ID: REQ-06	
Nome	Elaborazione immagini con account
Priorità	1
Versione	1.0
Note	Sarà possibile elaborare immagini caricando una propria immagine e selezionando il tipo di discromatopsia che si vuole applicare.

Sotto requisiti	
001	Se si ha fatto l'accesso sul sito l'immagine verrà salvata nel database e sarà possibile visualizzarla nell'history.
002	Le immagini si potranno elaborare anche senza fare l'accesso ma in questo caso non verranno salvate da nessuna parte.
003	Quando si selezionano i tipi di discromatopsia ci sarà una spunta per selezionarli tutti in automatico.

ID: REQ-07	
Nome	Protocollo di logging
Priorità	1
Versione	1.0
Note	Sul server dovrà essere presente un protocollo di logging per poter loggare delle informazioni importanti.

ID: REQ-08	
Nome	Possibilità di cambiare lingua
Priorità	1
Versione	1.0
Note	Sul sito sarà possibile cambiare lingua

ID: REQ-9	
Nome	Sito responsive
Priorità	1
Versione	1.0
Note	Il sito dovrà adattarsi alla grandezza dello schermo

ID: REQ-10	
Nome	Caricamento history in modo dinamico
Priorità	2
Versione	1.0
Note	La history non dovrà caricarsi tutta assieme ma mentre si scorre carica i nuovi valori.

ID: REQ-11	
Nome	Pagina o pop-up di informazioni
Priorità	2
Versione	1.0
Note	Nel sito dovrà essere presente una pagina o un pop-up che mostra un esempio per ogni tipo di discromatopsia per far capire all'utente in cosa consiste.

2.2.1 Spiegazione elementi tabella dei requisiti:

ID: identificativo univoco del requisito

Nome: breve descrizione del requisito

Priorità: indica l'importanza di un requisito nell'insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

Versione: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l'ultima versione, mentre le vecchie dovranno essere inserite nei diari.

Note: eventuali osservazioni importanti o riferimenti ad altri requisiti.

Sotto requisiti: elementi che compongono il requisito.

2.3 Use case

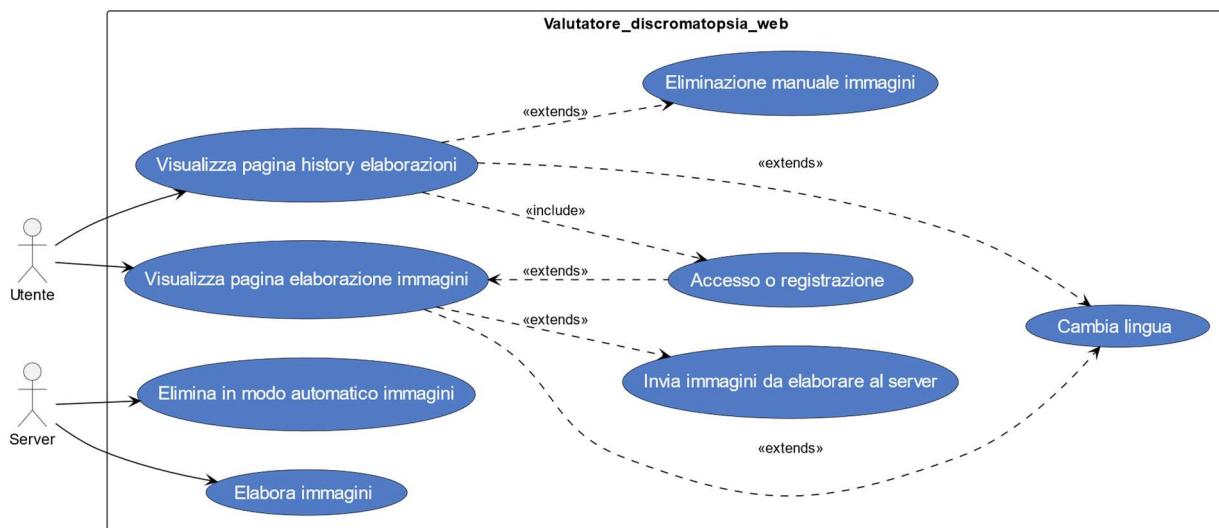


Figura 1 Use Case

2.3.1 Utente

L'utente all'interno del sito web potrà visualizzare la pagina delle elaborazioni delle immagini. Da questa pagina l'utente avrà la possibilità di inviare al server le immagini da elaborare. In più, volendo, può creare un account e eseguire l'accesso in modo da poter salvare le immagini che genera all'interno del database. Le immagini salvate saranno visibili all'interno dell'history dove, se si vuole, si potranno eliminare in modo manuale. Sia dalla pagina delle elaborazioni che dalla pagina dell'history sarà presente la possibilità di cambiare la lingua del sito.

2.3.2 Server

Il server invece avrà il compito di elaborare le immagini che riceve dall'utente con i vari tipi di discromatopsia. Inoltre avrà anche il compito di eliminare le immagini salvate all'interno del database dopo 30 giorni dall'elaborazione.

2.4 Pianificazione

Per questo progetto ho utilizzato il metodo waterfall. Il diagramma di Gantt è suddiviso in più sezioni ovvero: Analisi, Progettazione, Implementazione (Backend e Frontend) e Test+Debug. In più c'è la documentazione e i diari che vengono portati avanti in parallelo durante tutta la durata del progetto.

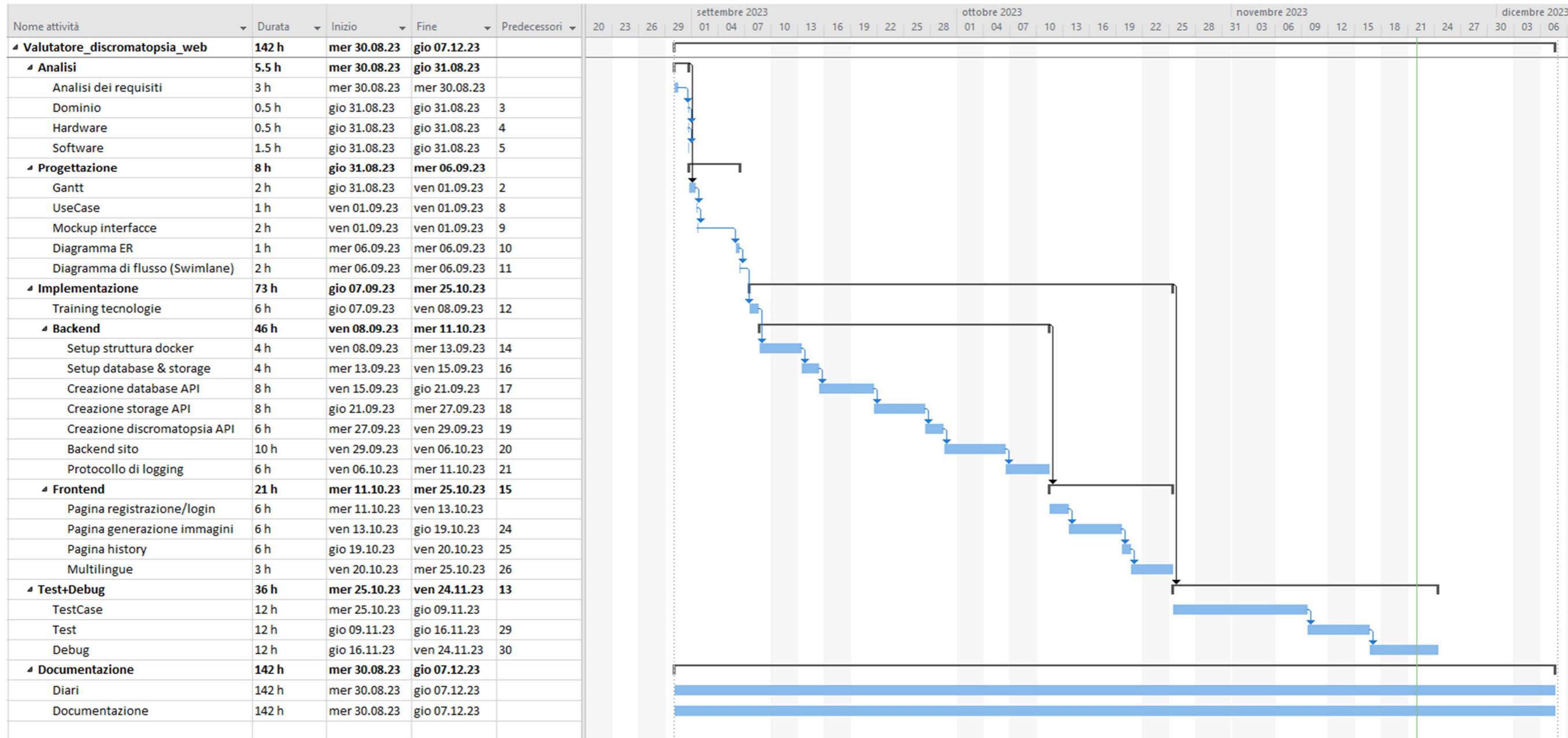


Figura 2 Gantt preventivo

2.5 Analisi dei mezzi

2.5.1 Software

- Visual Studio Code 1.78.2
- PlantUML VSCode Extension 2.17.5
- Remote – SSH VSCode Extension 0.102.0
- Microsoft Project 2019
- Microsoft Word 2019
- Microsoft Visio 2019
- VirtualBox 7.0.6
- Ubuntu Server 22.04
- MobaXTerm 22.1
- Postman 10.20.7
- Google Chrome 119.0.6045.200

2.5.2 Hardware

Il progetto è stato sviluppato su un pc fornito dalla scuola con le seguenti caratteristiche:

- Sistema Operativo: Windows 10 Enterprise
- Versione: 22H2
- Processore: Intel(R) Core (TM) i7-9700 CPU @ 3.00GHz
- RAM: 32.0 GB
- GPU: NVIDIA GeForce RTX 2060
-

In più, durante lo sviluppo ho dovuto utilizzare una macchina virtuale come server con le seguenti caratteristiche:

- Sistema Operativo: Ubuntu Server 22.04
- RAM: 4 GB
- Archiviazione: 70 GB
- Processori: 2 Core

3 Progettazione

3.1 Design dell'architettura del sistema

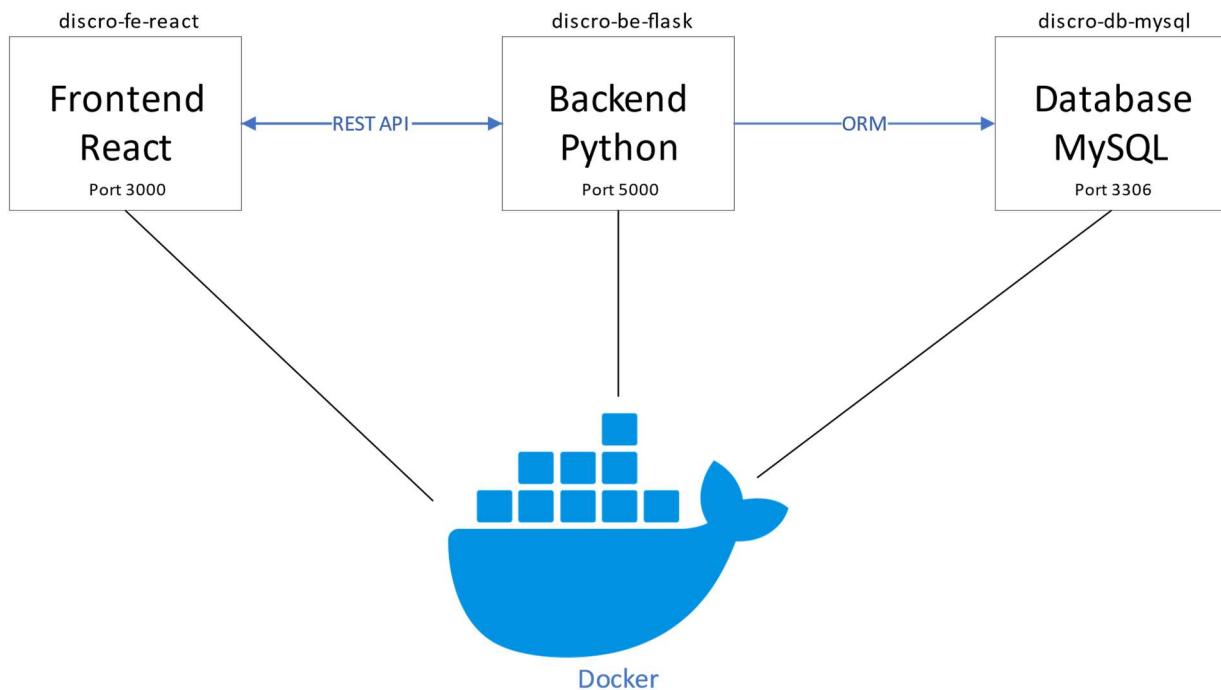


Figura 3 Schema di rete

Questo schema mostra come sarà strutturata l'applicazione. Infatti l'applicazione sarà composta da 3 container Docker (discro-fe-react, discro-be-flask e discro-db-mysql). Il frontend comunicherà con il backend tramite una REST API raggiungendo il container sulla porta 5000 e il backend sarà collegato al database grazie ad un ORM collegandosi sulla porta 3306. Il frontend sarà raggiungibile dagli utenti sulla porta 3000.

3.2 Design dei dati e database

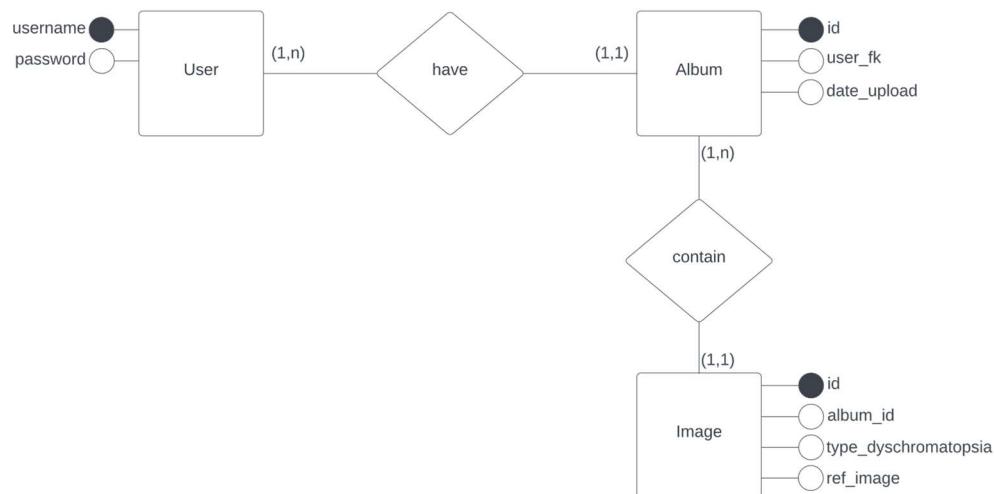


Figura 4 Diagramma ER

Qua sopra si può vedere il diagramma ER che mostra la struttura del database che verrà utilizzato. Sotto invece è presente lo schema logico del database. Il database infatti sarà composto solo da 3 tabelle: la tabella User, la tabella Album e la tabella Image.

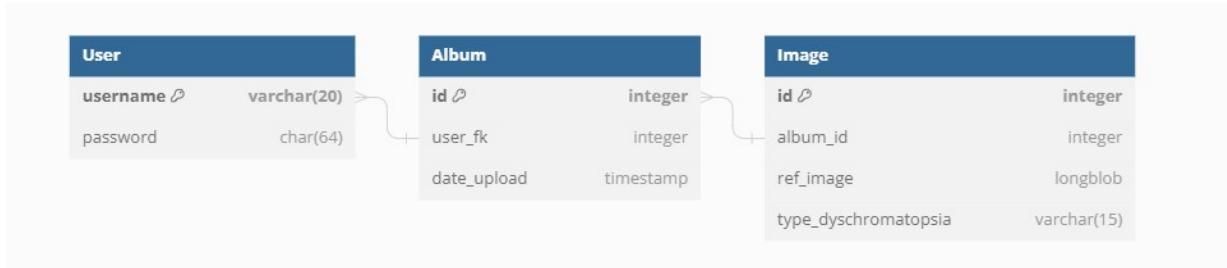


Figura 5 Schema logico

3.2.1 User

La tabella user conterrà solamente l'username dell'utente come chiave e la password. La password sarà salvata in un CHAR di lunghezza 64 perché l'algoritmo di hash che andrà ad utilizzare ritorna sempre 64 caratteri.

3.2.2 Album

La tabella album conterrà un suo id come primary key, l'id dell'utente a cui appartiene e la data di quando è stato creato per poter eliminare gli album quando sono più vecchi di 30 giorni.

3.2.3 Image

La tabella image invece conterrà come chiave un id, l'id del gruppo a cui appartiene l'immagine, l'immagine salvata in un longblob e il tipo di discromatopsia.

3.3 Design delle interfacce

I mockup che ho realizzato sono molto semplici, ma almeno ho già una bozza di come strutturare le varie pagine.

3.3.1 Pagina login

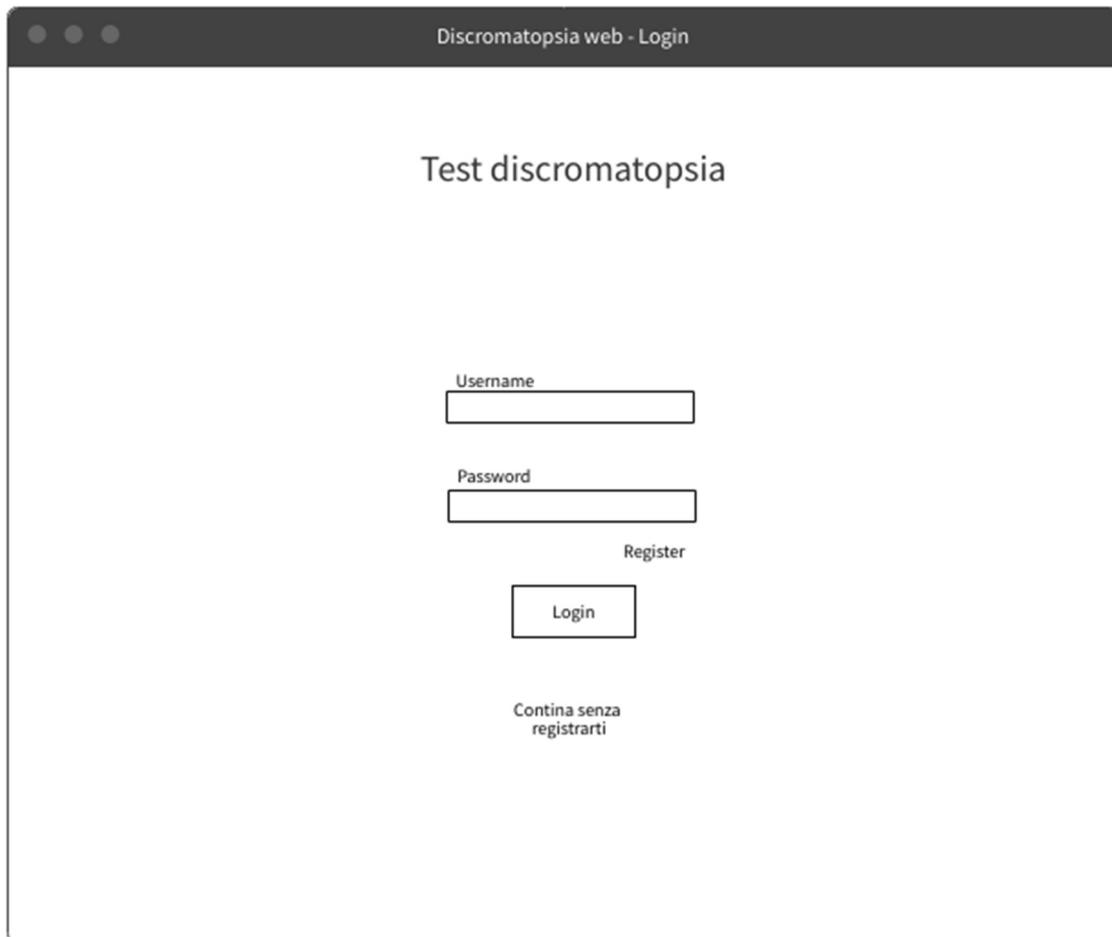


Figura 6 Mockup pagina login

La pagina di login è dove si può accedere al proprio account in modo da poter visualizzare le proprie foto generate in precedenza e poter salvare le foto che si generano. Nel caso non si ha ancora un account è presente un pulsante che porta alla pagina di registrazione. Se non si volesse accedere è presente anche un pulsante che ti porta alla pagina principale in modo da poter elaborare le foto senza salvarle.

3.3.2 Pagina registrazione

The mockup shows a registration form titled "Discromatopsia web - Register". It includes fields for "Username", "Password", and "Conferma Password" (Confirm Password), each with an associated input box. Below these fields are two buttons: "Login" and "Register". At the bottom left, there is a link labeled "Continua senza registrarti" (Continue without registering).

Figura 7 Mockup pagina registrazione

Nella pagina di registrazione si può creare il proprio account in modo da poter salvare e guardare in seguito le foto generate. Per registrarsi al sito è richiesto un nome utente e una password da ripetere due volte. Anche in questa pagina è presente il pulsante per andare alla pagina di login, nel caso si avesse già un account, oppure ti elaborare immagini senza salvarle.

3.3.3 Pagina generazione immagini

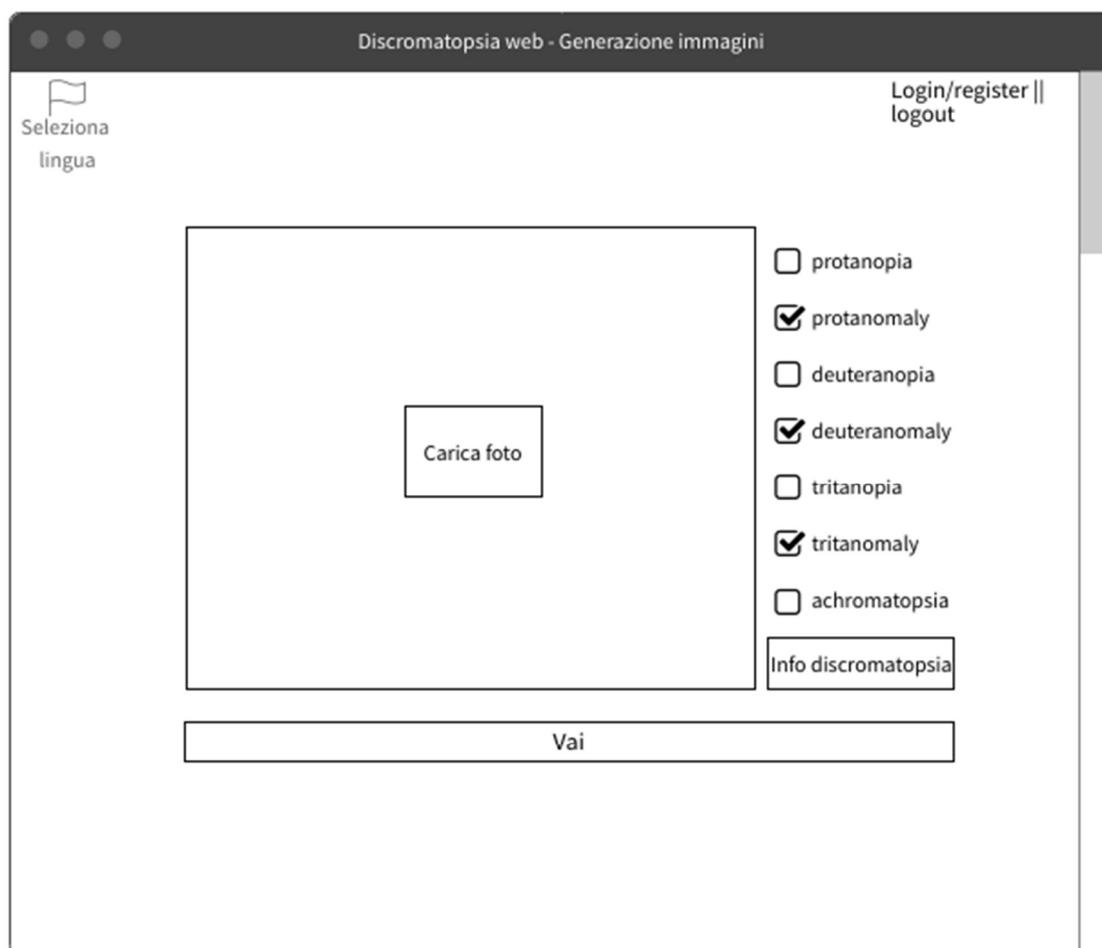


Figura 8 Mockup pagina principale

Questa è la pagina principale dove si può caricare una foto, selezionare uno o più tipi di discromatopsia e cliccare il pulsante per elaborare le foto. In questa pagina si può anche cambiare lingua e fare login/logout. In più sotto i vari tipi di discromatopsia c'è un pulsante che aprirà una finestra dove viene spiegato velocemente cosa sono i vari tipi di discromatopsia con degli esempi

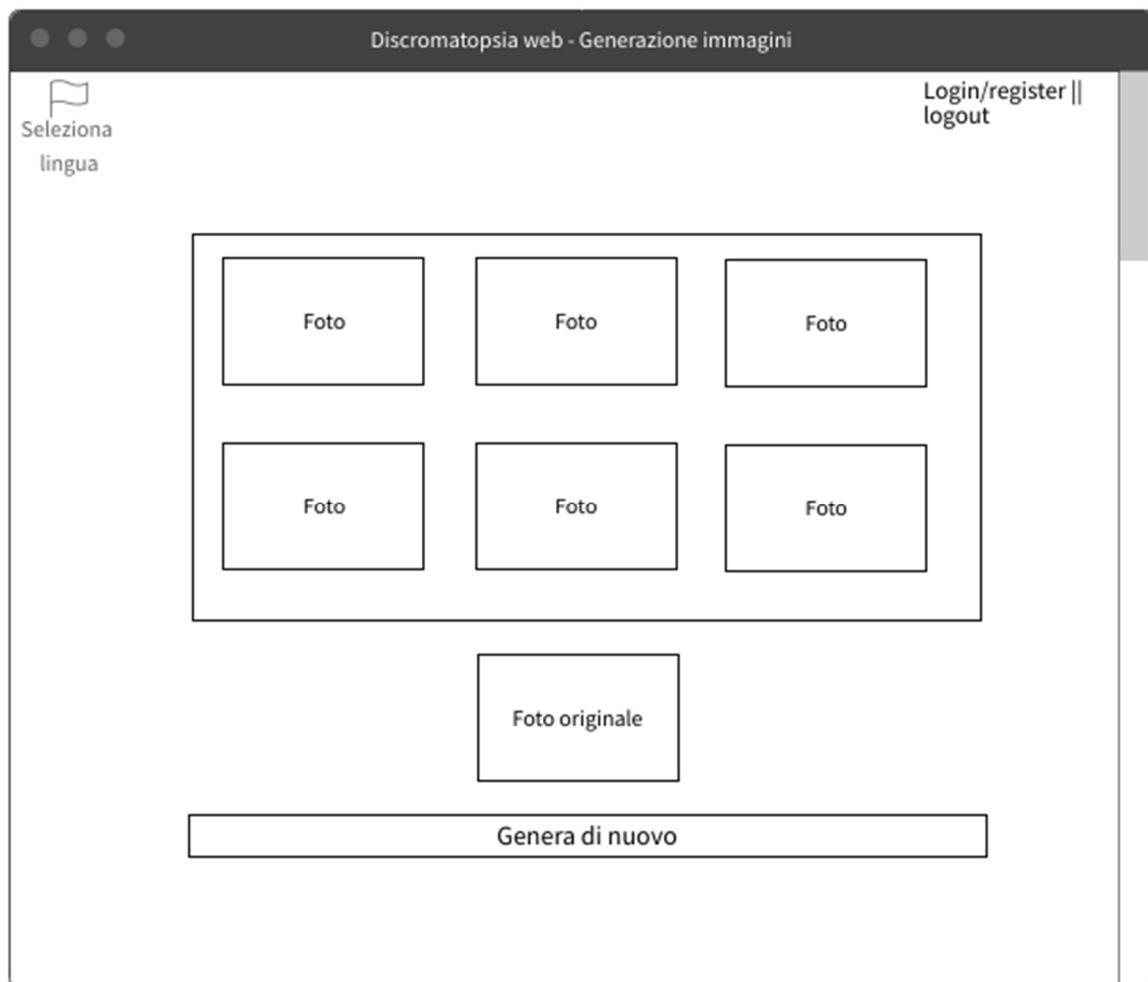


Figura 9 Mockup immagini mostrato dopo esser state generate

Una volta generate le immagini verranno mostrate a schermo prendendo il posto del form per caricare le immagini e i checkbox per selezionare il tipo di discromatopsia. In più comparirà anche un pulsante permetterà di elaborare una nuova immagine tornando allo stato iniziale della pagina.

3.3.4 Pagina history

The mockup shows a web application window titled "Discromatopsia web - Cronologia". On the left, there is a language selection dropdown labeled "Selezione lingua" with a flag icon. On the right, there is a "Login/register || logout" link. The main content area contains a table with three columns: "data", "foto generale", and "foto generate". The table has four rows. The first row shows data from 2023-09-07, with "foto originale" in the second column and "foto generate con tipo dis..." in the third. The second row shows data from 2023-09-08, with "foto originale" in the second column and "foto generate con tipo dis..." in the third. The third row shows data from 2023-09-09, with "foto originale" in the second column and "foto generate con tipo dis..." in the third. The fourth row is empty.

data	foto generale	foto generate
2023-09-07	foto originale	foto generate con tipo dis...
2023-09-08	foto originale	foto generate con tipo dis...
2023-09-09	foto originale	foto generate con tipo dis...

Figura 10 Mockup pagina history

Nella pagina dell'history (visualizzabile solo se si ha fatto l'accesso) si potranno vedere le foto generate negli ultimi 30 giorni comparate con la foto originale. Inoltre si potranno eliminare le foto vecchie in modo manuale. In più si potrà anche in questa pagina cambiare lingua. Per accedere a questa pagina bisogna scorrere verso il basso nel sito.

3.4 Design procedurale

3.4.1 Diagrammi di flusso

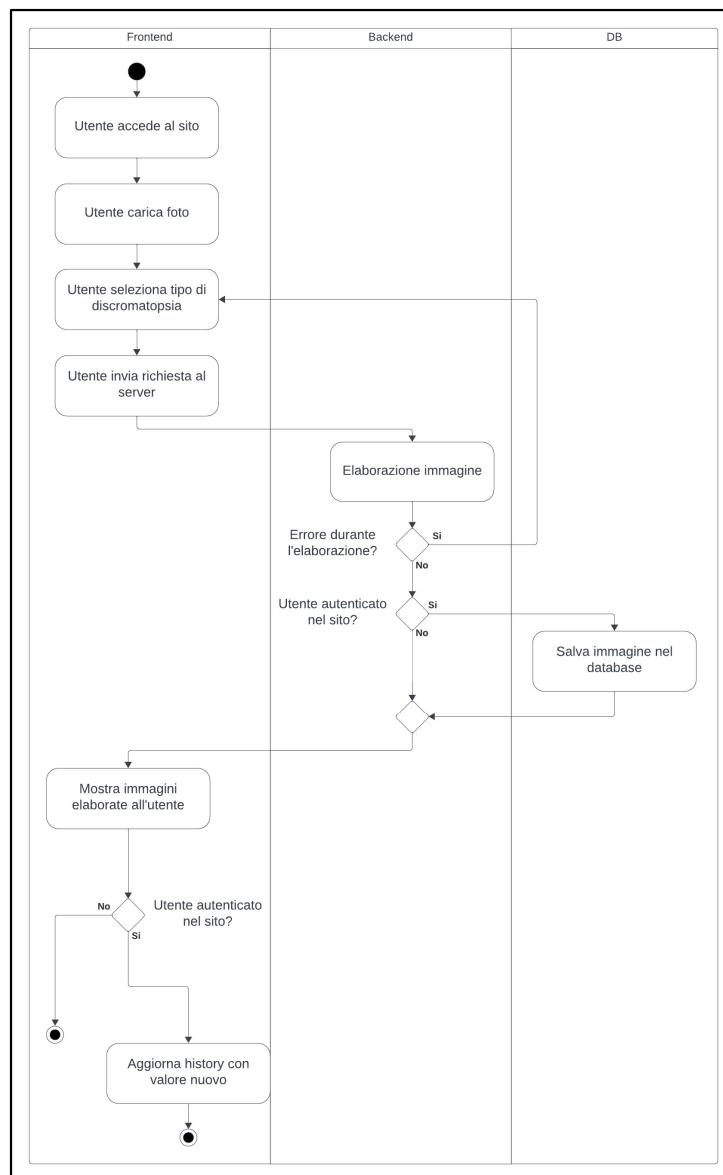


Figura 11 Swimlane generazione immagini

Questo primo schema mostra il processo con il quale vengono generate le immagini. Il risultato finale è quello di poter visualizzare a schermo le immagini generate con i tipi di discromatopsia selezionati e, nel caso si avesse fatto l'accesso all'interno del sito, che le immagini siano state salvate nel database. Quando le immagini vengono mostrate all'utente si dovrà aggiornare in automatico anche l'history (anche in questo caso solo se l'utente ha fatto l'accesso).

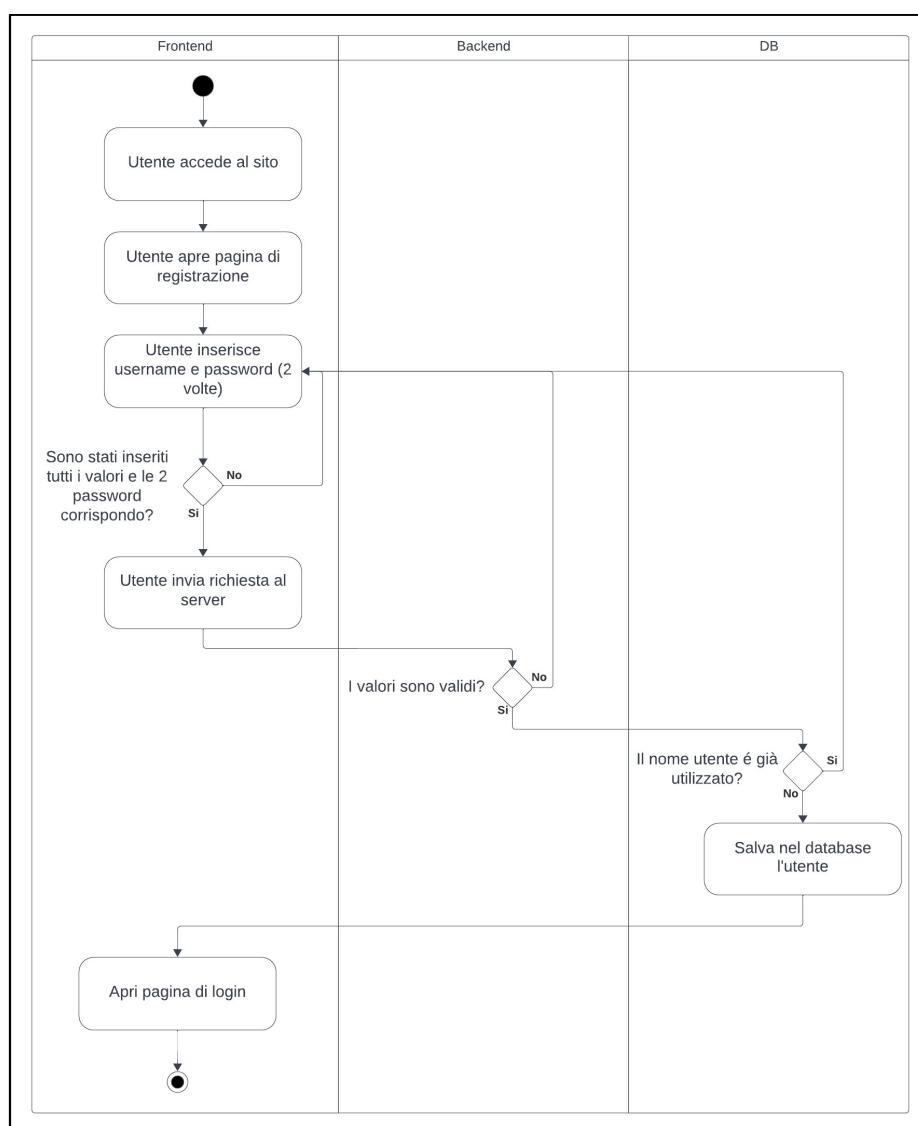


Figura 12 Swimlane registrazione utente

Questo schema mostra il processo con il quale un utente si registra. Se tutto funziona l'utente verrà portato alla pagina di login dove potrà effettuare l'accesso con il suo account appena creato. Ogni volta che verrà generato un errore il frontend lo gestirà per mostrarlo all'utente e bisognerà riinserire i valori non validi.

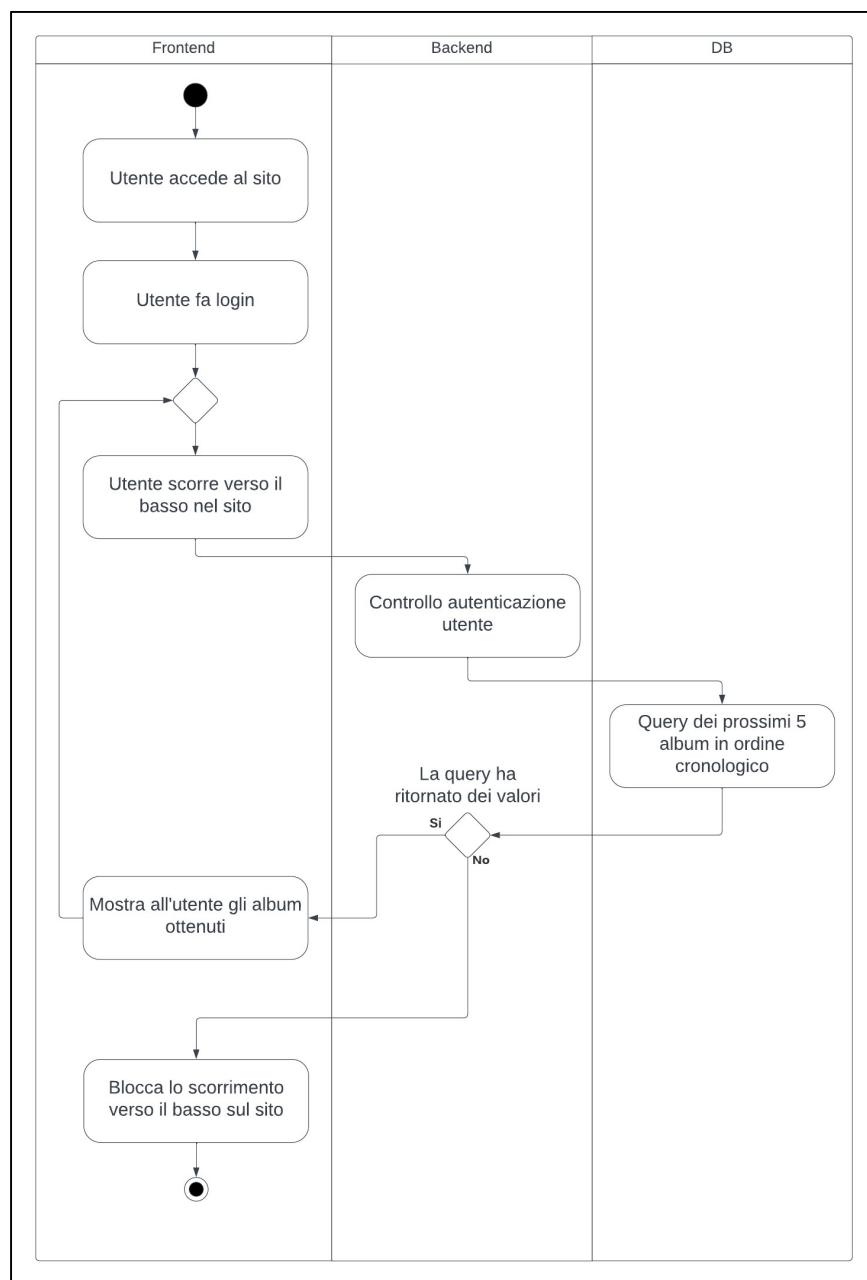


Figura 13 Swimlane visualizzazione history

Quest'ultimo schema mostra il funzionamento dell'history. Il risultato che si vuole ottenere è quello di poter visualizzare sul sito tutte le immagini generate negli ultimi 30 giorni. L'history funzionerà in modo dinamico quindi non vengono caricati tutti i valori assieme appena si apre il sito ma vengono caricati di 5 in 5 mentre si scorre verso il basso all'interno del sito. Quando tutti i valori saranno caricati non si potrà più scorrere verso il basso.

4 Implementazione

4.1 Struttura cartelle

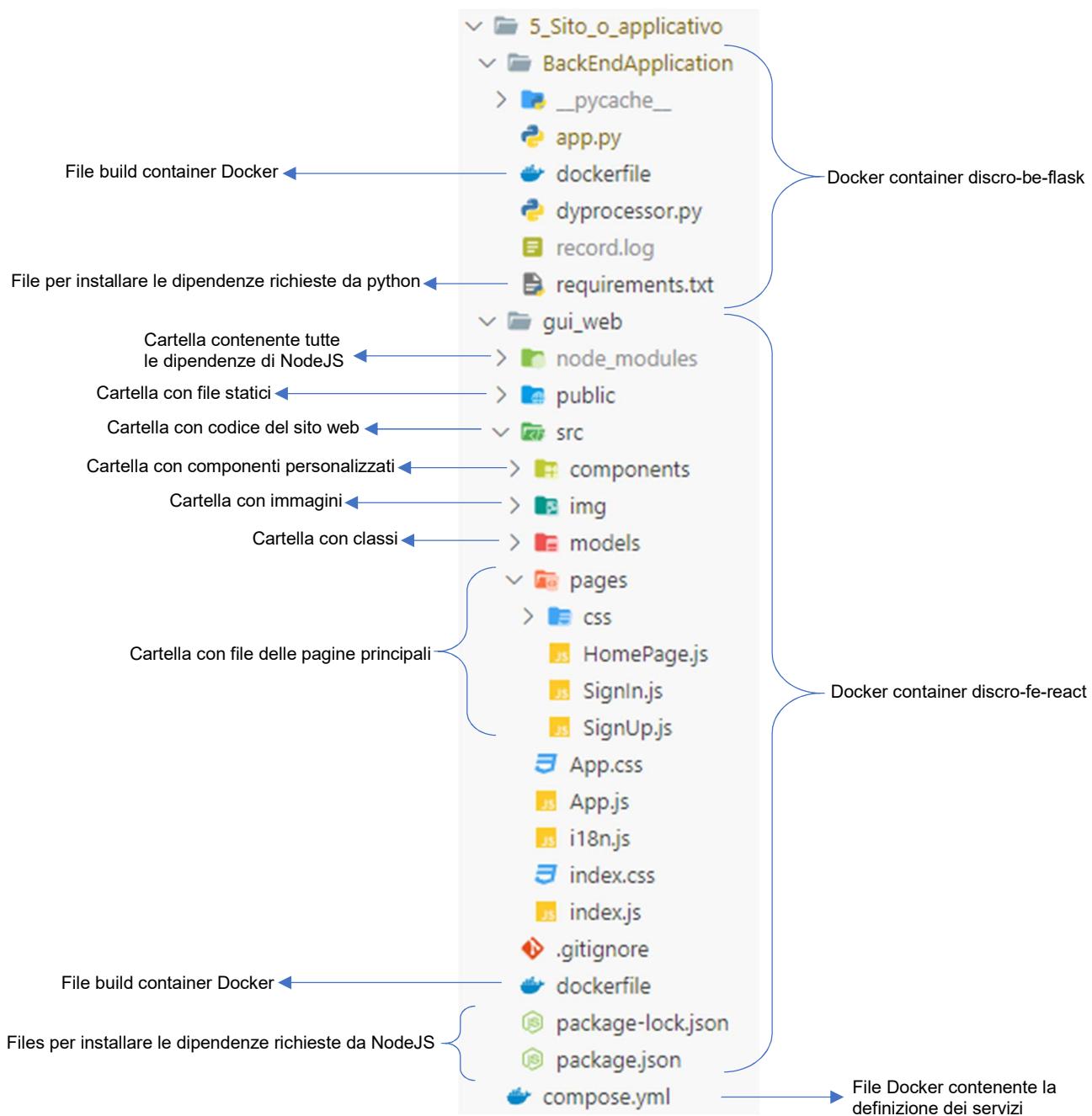


Figura 14 Struttura cartelle

4.2 Backend

Per il backend ho deciso di utilizzare python. Il compito principale del backend è quello di interfacciarsi con il database, dove verranno salvati gli utenti e le loro immagini generate, e quello di comunicare con il frontend grazie ad una REST API. Ho scelto di utilizzare python perché il progetto che mi è stato consegnato che elabora le immagini era fatto in python. La versione di python che ho utilizzato è la 3.10.6.

Con python ho utilizzato il framework Flask che mi ha permesso di creare la REST API.

Per lavorare con il database ho utilizzato la libreria SQLAlchemy, ovvero un ORM che permette di lavorare con molti tipi di database. Per il database ho deciso di utilizzare MySQL perché lo conosco già abbastanza bene.

4.2.1 Database

Per la creazione del database con SQLAlchemy per prima cosa bisogna configurare la stringa di connessione per il proprio database.

```
app.config['SQLALCHEMY_DATABASE_URI'] = os.getenv('DB_CONNECTION_STRING')
```

Per creare le tabelle bisogna dichiarare delle classi (una per tabella) che verranno poi trasformate da SQLAlchemy in tabelle all'interno del database.

4.2.1.1 User

```
class User(db.Model):
    username = db.Column(db.String(100), primary_key=True)
    password = db.Column(db.String(64).with_variant(CHAR(64), "mysql", "mariadb"))
    albums = db.relationship('Album', backref='user')

    def __repr__(self):
        return f'<User: {self.username}>'
```

Questa è la dichiarazione della tabella User. Questa tabella ha la proprietà username e password. Il campo albums serve da riferimento per poter poi creare una foreign key. Infine è presente la funzione __repr__ che serve per stampare un oggetto User.

Questa classe ha anche 3 funzioni che servono per gestire l'autenticazione della REST API ovvero una funzione per verificare la password, una funzione che crea un token d'autenticazione per l'utente e una funzione che verifica che i token siano validi.

4.2.1.2 Album

```
class Album(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    user_fk = db.Column(db.String(100), db.ForeignKey('user.username'))
    date_upload = db.Column(db.DateTime)
    images = db.relationship('Image', cascade="all,delete", backref='album')

    def __repr__(self):
        return f'<Album: {self.id}, User: {self.user_fk}, Created: {self.date_upload}>'
```

Questa è la dichiarazione della tabella Album. Questa tabella ha la proprietà id che è la chiave primaria ed è auto incrementale, la proprietà user_fk ovvero una foreign key che fa riferimento alla tabella User per sapere chi ha creato l'album e la proprietà date_upload per sapere quando viene creato l'album. In più anche in questa tabella viene dichiarato un riferimento per una foreign key chiamato images che ha impostato l'eliminazione in cascata in modo che se viene eliminato l'album vengono eliminate le immagini che ne fanno parte. Infine anche questa classe ha la funzione __repr__ per stampare l'oggetto.

4.2.1.3 Image

```
class Image(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    ref_image = db.Column(db.LargeBinary(length=(2**32)-1))
    type_dyschromatopsia = db.Column(db.String(15))
    album_id = db.Column(db.Integer, db.ForeignKey('album.id'))

    def __repr__(self):
        return f'<Image: {self.id}, Type: {self.type_dyschromatopsia}, Album: {self.album_id}>'
```

Questa è la dichiarazione della tabella Image. Questa tabella ha la proprietà id che è la primary key ed è auto incrementale, la proprietà ref_image che contiene l'immagine come longblob, la proprietà type_dyschromatopsia che contiene il tipo di discromatopsia che è stato applicato all'immagine e infine la proprietà album_id che fa riferimento alla proprietà id nella tabella album per determinare a che album appartiene l'immagine. In più anche questa classe ha la funzione __repr__ sempre per poter stampare l'oggetto. Questa classe ha anche una funzione che trasforma un elemento nel database in JSON in modo da poterlo inviare al frontend.

```
def convert_to_json(self):
    new_image = PIL_Image.open(io.BytesIO(self.ref_image))

    img_bytes = io.BytesIO()
    new_image.save(img_bytes, format=new_image.format)
    new_image = encodebytes(img_bytes.getvalue()).decode('ascii')
    return{
        "id": self.id,
        "ref_image": new_image,
        "type_dyschromatopsia": self.type_dyschromatopsia,
        "album_id": self.album_id
    }
```

4.2.1.4 Query

4.2.1.4.1 Select

Per eseguire un select sul database ho utilizzato il metodo query delle classi passando dei filtri alla query grazie alla funzione filter_by().

```
user = User.query.filter_by(username=username).one()
```

4.2.1.4.2 Insert

Per inserire valori nel database ho creato un oggetto di una tabella e poi l'ho aggiunto al database e infine ho fatto il commit per inserirlo effettivamente nel database. Finché non si esegue il commit il nuovo elemento non viene inserito all'interno del database.

```
newUser = User(username = username, password = password)
db.session.add(newUser)
db.session.commit()
```

4.2.1.4.3 Delete

Per rimuovere un oggetto dal database ho utilizzato il metodo delete() della sessione del database passando come parametro l'oggetto che dovevo eliminare e infine ho fatto il commit. Anche in questo caso finché non si esegue il commit le modifiche non vengono applicate.

```
album = Album.query.filter_by(id=id_user, user_fk=g.user.username).one()
db.session.delete(album)
db.session.commit()
```

4.2.2 REST API

Per far comunicare il backend e il frontend ho creato una REST API. Tutte le richieste ritornano dei JSON con all'interno almeno la proprietà "status" che può avere il valore "OK" se la richiesta ha avuto successo oppure "NOK" se c'è stato un errore. Se c'è un errore viene anche inviato un messaggio di errore.

4.2.2.1 Autenticazione

Per l'autenticazione della REST API ho utilizzato dei token salvati in memoria sul server. Il token viene inviato all'utente quando fa il login e viene salvato nei cookies. Ogni volta che l'utente farà una richiesta che necessita l'autenticazione verrà inviato al server anche il token in modo che possa autenticarsi.

```
def create_auth_token(self, expiration = 1000 * 3600 * 24 * 30):
    token = Serializer(app.config['SECRET_KEY'], expires_in=expiration)
    return token.dumps({ 'username': self.username })
```

Questa funzione viene utilizzata per creare un token. Il token viene criptato con una chiave segreta salvata nelle variabili d'ambiente. I token scadono dopo 30 giorni, dopodiché bisogna rieseguire l'accesso per creare un nuovo token.

```
@auth.verify_password
def verify_password(username_or_token, password):
    user = User.verify_auth_token(username_or_token)
    if not user:
        user = User.query.filter_by(username = username_or_token).first()
        if not user or not user.verify_password(password):
            return False
    g.user = user
    return True
```

Questa funzione invece viene utilizzata per controllare se l'autenticazione è valida. Per prima cosa viene controllato se il valore passato è un token valido, se non lo è viene controllato se il primo parametro è un username e il secondo una password. Se il token esiste o se username e password sono corretti viene ritornato True e le funzioni che necessitano l'autenticazione vengono eseguito, se invece l'autenticazione non è valida questa funzione ritorna False e le funzioni che necessitano l'autenticazione non verranno eseguite e verrà inviata come risposta al client un errore 401 UNAUTHORIZED.

```
@app.route('/generate', methods=['POST'])
@auth.login_required
def generateImages():
```

Per specificare che una route necessita dell'autorizzazione bisogna impostare il decoratore @auth.login_required alla funzione della route. Un decoratore è un'estensione di una funzione in python.

4.2.2.2 Routes

4.2.2.2.1 Login

La route login serve per fare l'accesso sul sito. Questa route viene richiamata dalla pagina di Sign in e semplicemente richiede l'autenticazione e se le credenziali sono valide ritorna il token assegnato all'account in modo che per le prossime richieste non bisogna continuare a mandare il nome utente e la password ma si può utilizzare il token d'autenticazione.

4.2.2.2.2 Createuser

Questa route invece serve per creare un utente sul sito. Questa route viene richiamata dalla pagina di Sign up e viene controllato se l'username è valido e se non è già utilizzato. La password invece viene validata con una regular expression.

```
PSW_PATTERN = re.compile("^(?=.*[0-9])(?=.*[!@#$%^&*])[a-zA-Z0-9!@#$%^&*]{6,}$")
```

Questa regular expression controlla che il valore passato dall'utente abbia almeno un numero, almeno un carattere speciale e che sia lungo almeno 6 caratteri.

Se è tutto valido la password viene hashata con lo SHA-256 e l'utente viene salvato nel database. Una volta registrati il sito porterà l'utente alla pagina di login per poter accedere al sito. In caso i valori non siano validi o già utilizzati verranno inviati al client dei messaggi d'errore che verranno gestiti dal sito. Lo schema qua sotto mostra tutti i controlli vengono effettuati dal backend e gli eventuali messaggi d'errore che vengono ritornati al frontend. Se non viene inviato alcun messaggio d'errore viene creato un nuovo utente all'interno del database.

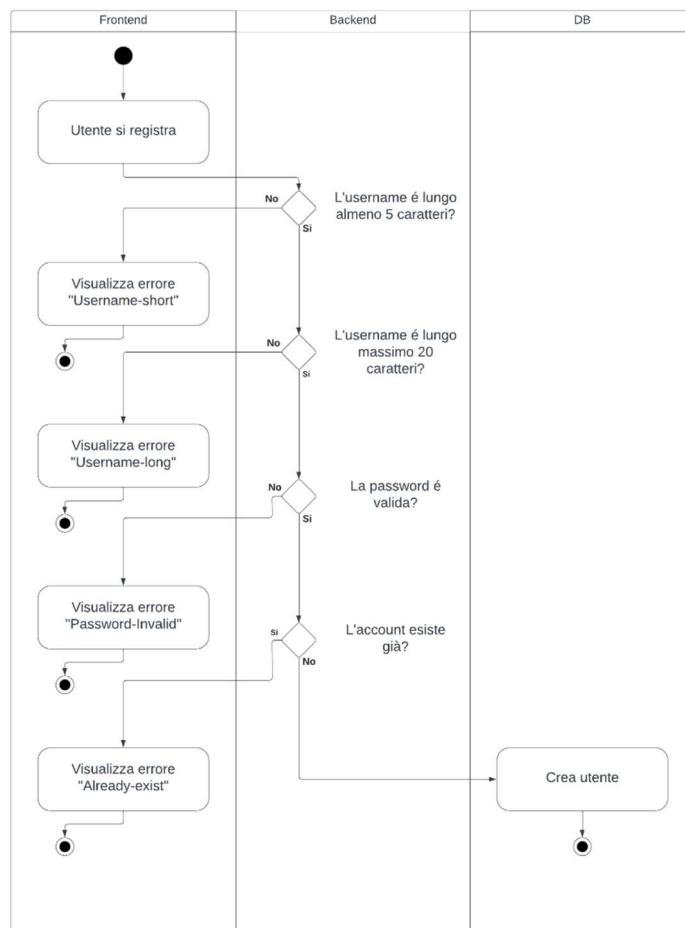


Figura 15 Swimlane gestione errori

4.2.2.2.3 Generate

Questa route utilizza il metodo POST e serve per elaborare le immagini nel caso si fosse autenticati. Infatti in cima alla route è presente il decoratore @auth.login_required. Questa route riceve come parametri un'immagine e un array contenente dei numeri. Ogni numero corrisponde ad un tipo di discromatopsia e quelli disponibili sono dallo 0 al 6. L'immagine ricevuta viene convertita in 2 formati: come PIL_Image per poi mandarla alla classe che elabora le immagini, e in bytes per poterla salvare all'interno del database come longblob. Infatti prima di andare ad elaborare le immagini viene creato nel database un nuovo album e viene inserita nel database l'immagine originale. Infine viene percorso tutto l'array contenente i tipi di discromatopsia e le immagini vengono generate una ad una. Le immagini elaborate vengono convertite in base64 per poi ritornarle al client in modo da mostrarle sul sito e vengono anche salvate all'interno del database.

4.2.2.2.4 Generatenouseraccount

Questa route è praticamente uguale alla route "generate" ma non richiede l'autenticazione e non viene salvato niente sul database. Questa route viene richiamata nel caso non si avesse fatto l'accesso sul sito.

4.2.2.2.5 Remove

Questa route serve per eliminare un proprio album. Anche questa route necessita dell'autenticazione e utilizza il metodo DELETE. Come parametro questa route necessita l'id dell'album che si vuole rimuovere. Per eliminare un album bisogna anche esserne il proprietario sennò verrà ritornato un errore.

```
try:  
    album = Album.query.filter_by(id=id_user, user_fk=g.user.username).one()  
    db.session.delete(album)  
    db.session.commit()  
    return jsonify({"status": STATUS_OK})  
except:  
    return jsonify({"status": STATUS_NOK, "message": "Impossible-remove"})
```

Come si vede dal codice, viene fatta una query controllando l'id dell'album e se lo user che ha creato l'album è lo stesso che ha fatto la richiesta.

4.2.2.2.6 History

La route history è la route che permette di caricare l'history delle elaborazioni. Questa route utilizza il metodo GET e necessita come parametro il numero della pagina per il caricamento dinamico e necessita anche dell'autenticazione. In questa route viene fatta una query per prendere tutti gli album appartenenti ad un utente e vengono ordinati in base alla data di creazione dal più recente al meno recente. Questa query utilizza anche la funzione paginate() che permette di suddividere il risultato di query in pagine passandogli l'indice della pagina alla quale si vuole accedere e quanti elementi deve contenere la pagina. Questa funzione mi serve per poter fare in modo che l'history si carichi in modo dinamico mentre si scende nella pagina e non vengono presi tutti i dati subito.

```
albums = Album.query.filter_by(user_fk=username)
.order_by(desc(Album.date_upload))
.paginate(page=page, per_page=per_page)
```

Una volta che si ottengono gli album vengono percorsi e per ogni album si prendono le foto per poi ritornarle al client.

```
for album in albums:
    images = Image.query.filter_by(album_id=album.id)
    img_arr = []
    for img in images:
        img_arr.append(img.convert_to_json())
    history.append({"images": img_arr, "date_upload": album.date_upload})
return jsonify({"status": STATUS_OK, "history":history})
```

Nel caso in cui non ci siano più valori nell'history dell'utente verrà ritornato un messaggio di errore per farlo sapere al frontend.

```
return jsonify({"status": STATUS_NOK, "message": "No-more-history-values"})
```

4.2.3 Scheduled Task

Nel backend è presente anche una scheduled task che serve per fare l'eliminazione automatica degli elementi più vecchi di 30 giorni. Per farla ho utilizzato la libreria di python APScheduler e poi ho dichiarato la mia scheduled task per eseguire una funzione ogni giorno.

```
scheduler.add_job(id="remove", func=remove_old, trigger="interval", days=1)
scheduler.start()
```

La funzione remove_old esegue una query per ottenere tutti gli album presenti nel database e poi controlla se sono passati più di 30 giorni da quando è stato inserito nel database con l'attributo date_upload. Se è più vecchia vengono eliminate dal database insieme alle immagini che contiene grazie all'eliminazione in cascata impostata alla dichiarazione della tabella.

```
albums = Album.query.all()
limit = datetime.today() - timedelta(days=30)
for alb in albums:
    if alb.date_upload < limit:
        db.session.delete(alb)
db.session.commit()
```

Per qualche motivo però la scheduled task non parte all'interno del container Docker quindi ho dovuto creare una route che richiama questa funzione in modo che nell'applicazione è comunque presente la funzionalità di rimuovere i valori vecchi, anche se non parte in modo automatico.

```
@app.route('/removeold', methods=['DELETE'])
@auth.login_required
def remove_old_route():
    remove_old()
    return jsonify({"status": STATUS_OK})
```

La route non viene richiamata all'interno del sito ma si può richiamare con un curl.

```
curl -u "admin:Password&1" http://192.168.56.50:5000/removeold -X DELETE
```

4.3 Frontend

Per il frontend ho deciso di lavorare con React ovvero una libreria di Javascript. Ho scelto di lavorare con React perché ci avevo già lavorato in passato e mi ci trovo bene. La versione di React è la 18.2.0. Insieme a React ho deciso di lavorare anche con la libreria di design Antd. Questa libreria offre dei componenti già pronti per poter creare un bel design. La versione di Antd è la 5.9.2.

4.3.1 Pagine e navigazione

Nel sito web sono presenti 4 pagine: la pagina per accedere, la pagina per registrarsi, la pagina principale dove si possono elaborare le immagini e vedere l'history e una pagina d'errore che viene mostrata se si prova ad accedere ad una route non esistente.

Per poter navigare tra le pagine ho utilizzato la libreria “react-router-dom” che permette di dichiarare le routes del frontend. Infatti nel componente App, che è il componente principale che contiene tutto il sito web, viene dichiarato su quale route aprire quale componente.

```
return (
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<HomePage />} />
      <Route path="/login" element={<SignIn />} />
      <Route path="/register" element={<SignUp />} />
      <Route path='*' element={<NotFound />} />
    </Routes>
  </BrowserRouter>
);
```

Come si vede dal codice sovrastante se si prova accedere ad una route che non è stata dichiarata verrà aperta la pagina NotFound dove si potrà tornare alla pagina principale.

Per potersi spostare tra le pagine ho utilizzato il componente “Link” che permette di cambiare route del sito cliccandoci sopra.

```
<Link to="/">
  <label>Continue as guest</label>
</Link>
```

4.3.1.1 SignIn e SignUp

Le pagine SignIn e SignUp sono molto simili a livello grafico, l'unica differenza è che la pagina di SignUp ha un campo di testo di più nel form per inserire la conferma della password. Per poter accedere a queste pagine ci sono dei pulsanti nell'homepage nel caso non si fosse già registrati. Se si ha già fatto l'accesso invece dei due pulsanti ci sarà il pulsante di logout.

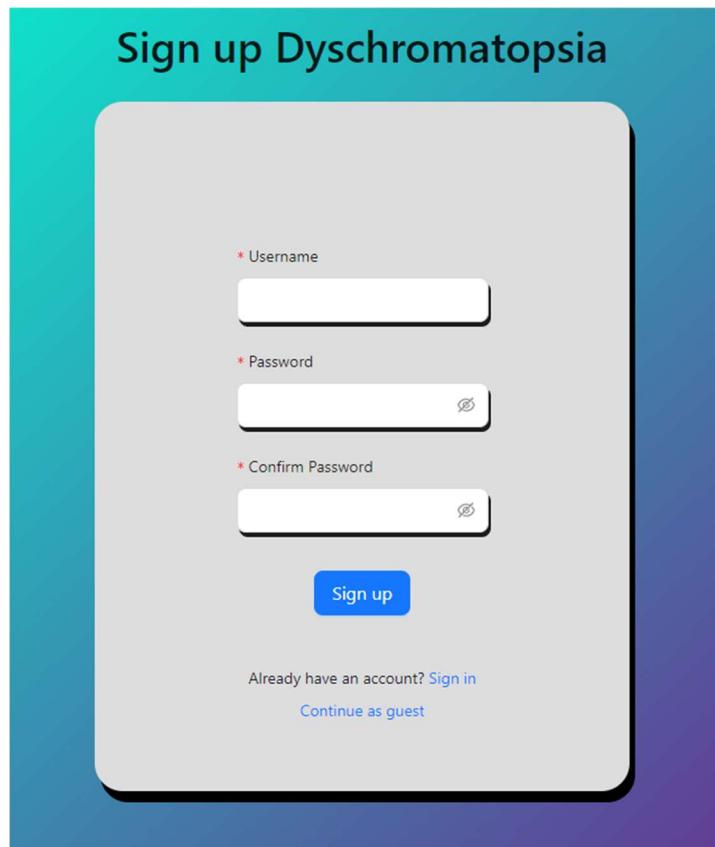


Figura 16 Pagina SignUp

Per registrarsi al sito web basta inserire un nome utente di minimo 5 caratteri, massimo 20 caratteri e che non sia già utilizzato e inserire una password 2 volte. Nel caso una di queste condizioni per l'username non verrà soddisfatta, apparirà a schermo una notifica che mostra l'errore e il campo di testo dell'username verrà svuotato.

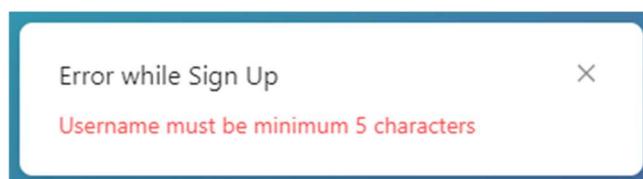


Figura 17 Notifica errore username

Anche la password ha delle condizioni per essere accettata: deve avere almeno 6 caratteri, almeno un numero e almeno un carattere speciale. Anche in questo caso, se le condizioni non vengono soddisfatte, apparirà una notifica con scritto un altro messaggio d'errore.

Nella parte bassa del form sono presenti anche due link che permettono di andare nella pagina di SignIn nel caso si avesse già creato un account oppure nell'Homepage nel caso non si volesse accedere al sito.

4.3.1.2 Homepage

In cima all'Homepage è presente un header dove si potrà cambiare lingua e, come detto in precedenza, sono presenti i pulsanti per accedere al sito oppure per fare logout.

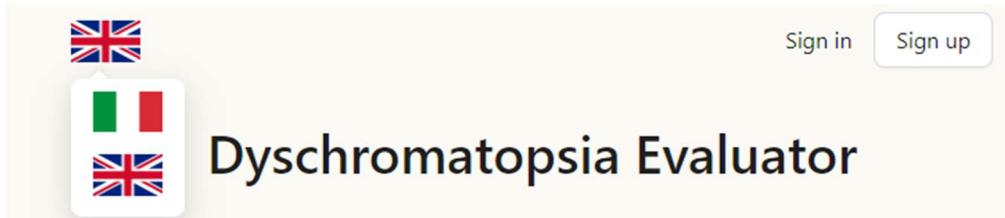
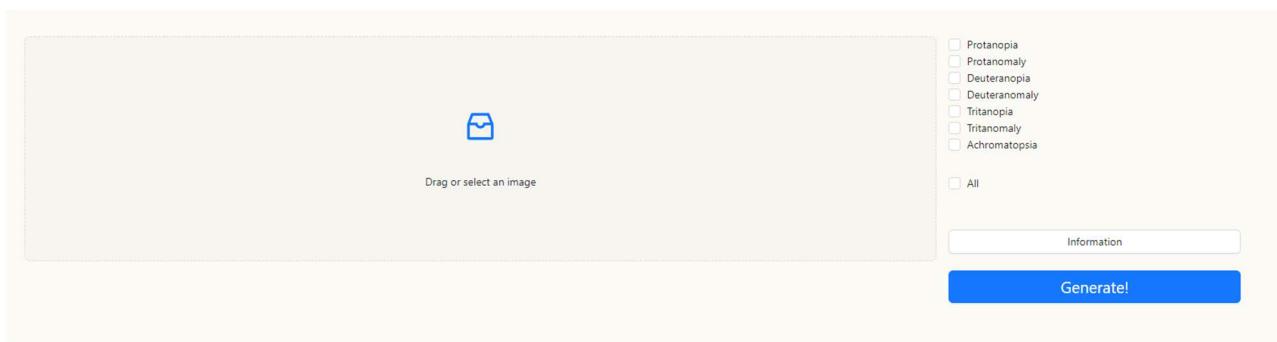


Figura 18 Header Homepage

Nella parte centrale dell'Homepage è presente un drag & drop per poter caricare l'immagine da elaborare. Quando si seleziona un file viene controllato se il file è un'immagine (i file accettati sono i png, jpg e jpeg), in caso contrario non verrà accettato e verrà mostrato un messaggio di errore. Nel caso il file fosse valido viene salvato in una variabile in base 64 e viene mostrato al posto del drag & drop in modo che si capisca che la foto è stata accettata.



The interface includes a large drag & drop area with a camera icon, a placeholder text "Drag or select an image", a list of checkboxes for discromatopsia types (Protanopia, Protanomaly, Deutanopia, Deutanomaly, Tritanopia, Tritanomaly, Achromatopsia), a general "All" checkbox, an "Information" input field, and a prominent blue "Generate!" button.

Figura 19 Homepage

Al lato del drag & drop sono presenti dei checkbox per selezionare quale tipo di discromatopsia applicare all'immagine. Selezionando il checkbox "All" gli altri checkbox verranno disattivati e l'immagine verrà elaborata con tutti i tipi di discromatopsia. Quando si clicca il tasto "Generate!" la foto e i tipi di discromatopsia verranno inviati al server (verrà spiegato dopo in che modo) e al posto del drag & drop ci sarà un'animazione di caricamento finché non si riceverà risposta dal server. Appena il server risponde le immagini elaborate verranno mostrate a schermo con scritto quale tipo di discromatopsia è stato applicato.

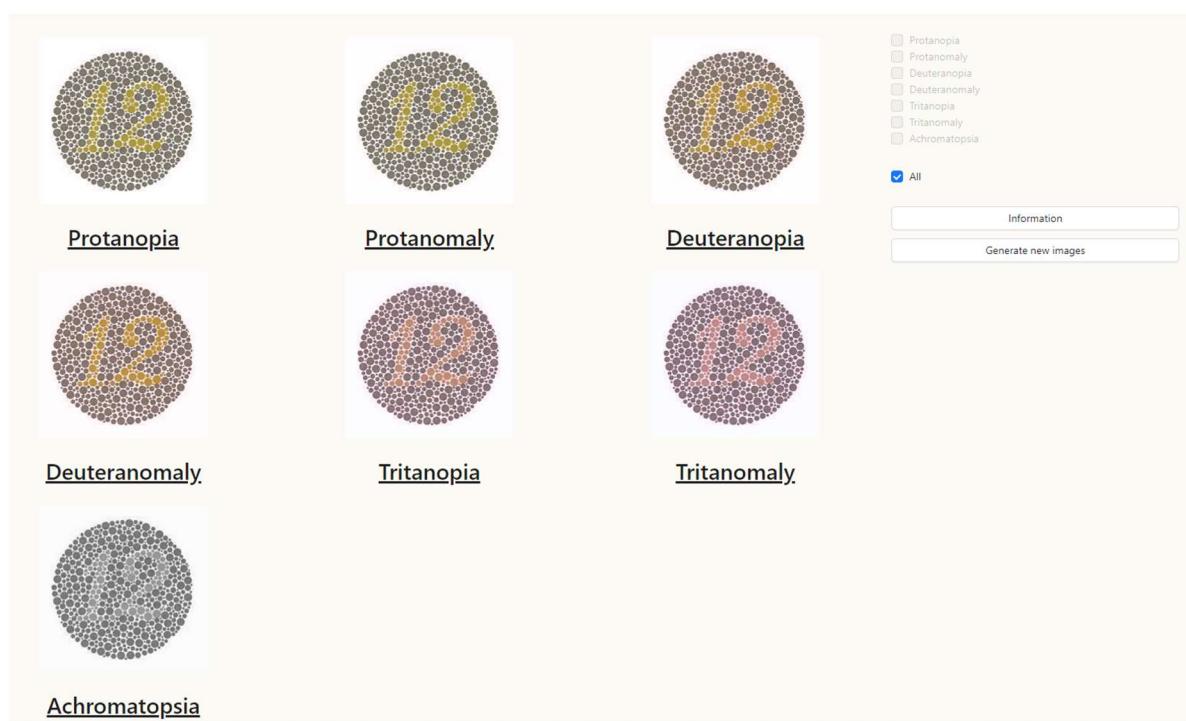


Figura 20 Immagini generate mostrate a schermo

In più, una volta che vengono mostrate le immagini, il pulsante “Generate!” sparirà e sotto al pulsante “Information” comparirà un pulsante “Generate New images” che permette di far riapparire il drag & drop e i checkbox vengono riattivati tutti e deselectivati tutti.

4.3.1.2.1 History

L’history si trova sotto al form per elaborare le immagini e sarà mostrata solo nel caso si avesse fatto l’accesso al sito. Altrimenti, se si scorre in basso, uscirà la seguente scritta.

Sign In for save your images in history

Figura 21 Messaggio al posto dell’history senza accesso

Se invece si ha fatto l’accesso ma non si ha ancora generato delle immagini uscirà un altro messaggio.

History

There aren’t values in your history

Figura 22 Messaggio se non si ha ancora generato nulla

Se invece si hanno dei valori all’interno dell’history compariranno degli elementi con l’immagine originale e la data della generazione.



Figura 23 Valore history

Quando si clicca su questo elemento si aprirà e mostrerà tutte le immagini elaborate con scritto sopra il tipo di discromatopsia applicato.

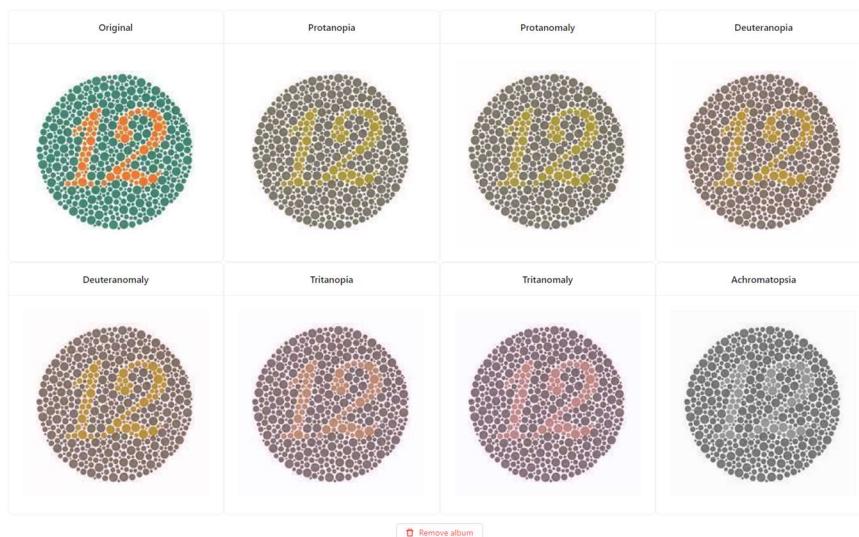


Figura 24 Contenuto valore nell'history

In più, volendo, si può cliccare il pulsante in basso che permette di eliminare una generazione di immagini in modo manuale.

Inoltre, come detto in precedenza nella parte del backend, l'history viene caricata in modo dinamico per evitare che vada a leggere tutti i dati generati dall'utente subito, ma che vada a leggerli mentre si scorre la pagina. Per farlo ho utilizzato un calcolo che controlla se si è arrivati in fondo alla pagina, incrementa di uno il numero della pagina e invia una nuova richiesta al server per i prossimi valori. Nel mentre all'utente viene mostrato un caricamento.

```

const windowHeight = "innerHeight" in window ? window.innerHeight :
  document.documentElement.offsetHeight;
const body = document.body;
const html = document.documentElement;
const docHeight = Math.max(body.scrollHeight, body.offsetHeight,
  html.clientHeight, html.scrollHeight, html.offsetHeight);
const windowBottom = windowHeight + window.pageYOffset;
  
```

Se la variabile windowBottom sarà maggiore o uguale a docHeight vado a fare richiesta dei nuovi valori al server come spiegato sopra.

4.3.1.2.2 Info

Quando si clicca il pulsante “Information” si aprirà una finestra che contiene un esempio per ogni tipo di discromatopsia in modo da fare capire all’utente cosa fanno i vari filtri.

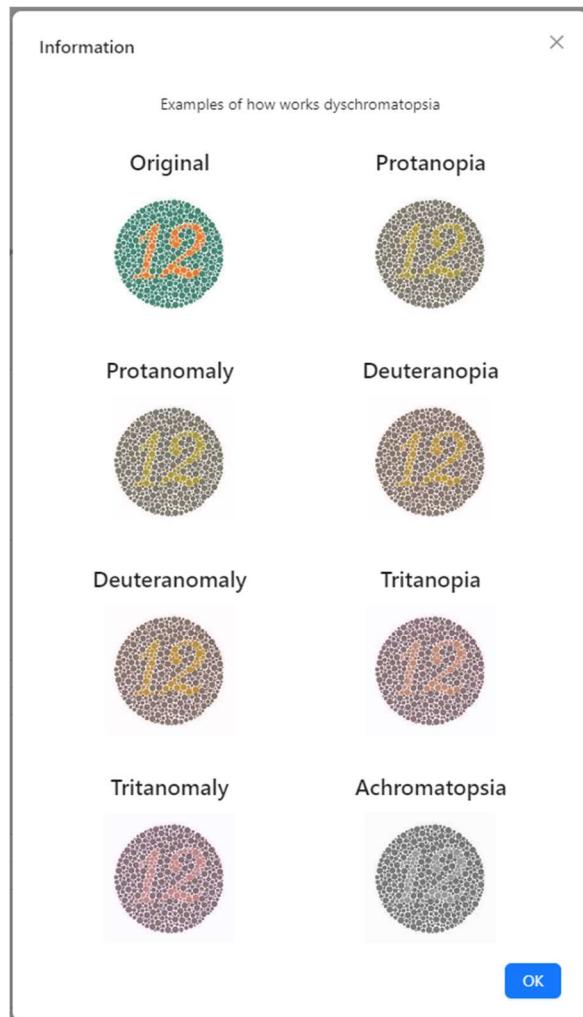


Figura 25 Modal informazioni

4.3.1.3 Error Page

La pagina di errore si aprirà se si prova ad accedere ad una route non dichiarata.

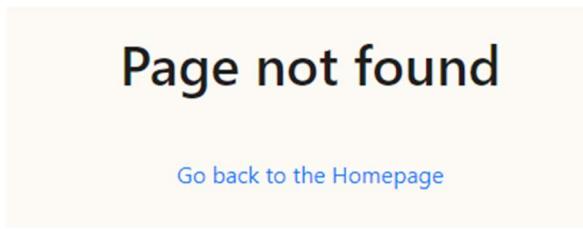


Figura 26 Pagina d’errore

In questa pagina è presente solamente un link per poter tornare all’Homepage.

4.3.2 Comunicazione con backend

Per comunicare con il server utilizzo la libreria “axios”. Questa libreria permette di fare delle richieste http.

```
axios({
  method: "post",
  url: process.env.REACT_APP_SERVER_URL + "/createuser",
  data: bodyFormData,
  headers: { "Content-Type": "multipart/form-data" },
})
.then((response) => {
```

Dei vantaggi che ha axios rispetto al fetch (funzione di Javascript già integrata per fare le richieste http) è che è supportato da più browser, converte in automatico le risposte in JSON e ha già una sua gestione degli errori.

4.3.2.1 Autenticazione

Per l'autenticazione delle richieste bisogna aggiungere un parametro nella richiesta con axios chiamato “auth”. Questo oggetto permette di inviare un oggetto con attributi “username” e “password”.

```
axios({
  method: "get",
  url: process.env.REACT_APP_SERVER_URL + "/login",
  headers: {
    "Content-Type": "multipart/form-data",
  },
  auth: {
    username: username,
    password: password
  }
})
```

Se invece mi devo autenticare con un token metto come valore dell'username il token e come password metto una stringa vuota perché tanto non verrà controllata.

4.3.2.2 Gestione errori

Per gestire i messaggi di errore che ritorna il backend ho creato la classe `ErrorHandler` che contiene una funzione `handle()` che passandogli il messaggio d'errore ritorna la stringa traducibile corrispondente da mostrare all'utente.

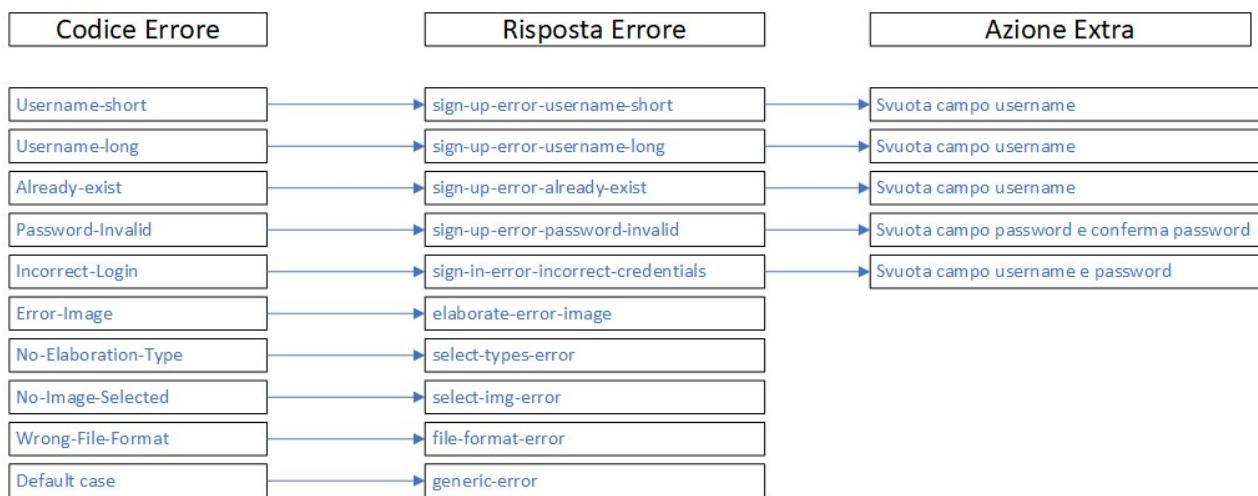


Figura 27 Schema gestione errori

In più si può passare un parametro extra ovvero il form che scatena l'errore perché certi errori necessitano di rimuovere dei valori dal form. Prima di andare a rimuovere i valori dal form controllo che il parametro `form` sia stato passato alla funzione grazie alla sintassi `<condizione> && <se vero>`. In questo caso la condizione è l'oggetto `form` e se ha un valore assegnato ritorna `true`, se invece non ha un valore ritorna `false` e quindi non viene chiamata la funzione per rimuovere i valori dal form.

```
form && form.resetFields(["username"]);
```

4.3.3 Multilingua

Per poter cambiare lingua all'interno della pagina utilizzo la libreria “`i18n`”. Questa libreria permette di scrivere i testi da mostrare all'utente in dei file JSON (uno per lingua) e con una funzione scegliere da quale JSON leggere i testi da mostrare.

Quando la lingua viene cambiata viene salvato anche un cookie che contiene la lingua che è stata selezionata in modo che quando si riapre la pagina è ancora selezionata la lingua impostata in precedenza.

```
i18n.changeLanguage(lng)
cookies.set('lng', lng, {
  path: '/',
  expires: new Date(new Date().getTime() + 1000 * 3600 * 24 * 30),
});
```

Per accedere al testo del JSON bisogna utilizzare un'altra funzione passandogli come parametro il nome della proprietà all'interno del JSON.

```
t("title")
```

Entrambi i JSON devono contenere le stesse proprietà in modo che in base alla lingua che si seleziona verrà mostrato il testo nella lingua corretta.

4.3.4 Responsive

Per rendere il sito responsive ho creato un hook di React personalizzato. Questo hook mi permette di accedere alla dimensione della finestra in ogni momento e se necessario vado a modificare la disposizione dei componenti per rendere il sito responsive.

```
const useResize = () =>{
  const [innerWidth, setInnerWidth] = useState(window.innerWidth);
  useEffect(() =>{
    const resize = () =>{
      setInnerWidth(window.innerWidth);
    }
    window.addEventListener("resize", resize)
    return () => {
      window.removeEventListener('resize', resize)
    }
  }, [])
  return [innerWidth];
}
```

Questo hook aggiunge un listener che controlla quando viene cambiata la dimensione della finestra e, quando succede, richiama una funzione che salva il valore della larghezza della finestra.

```
style={{width: innerWidth > 768 ? "32%" : "100%}}}
```

Per esempio in questo caso imposto la larghezza di un div controllando con un operatore terziario la larghezza dello schermo.

4.4 Docker

Per creare le tre applicazioni ho utilizzato docker compose che sfrutta il file compose.yml per poter far partire più container docker assieme definendo dei servizi. Per quest'applicazione ci sono tre servizi: il Backend, il Frontend e il Database.

4.4.1 Docker backend

Il servizio chiamato backend è il container che contiene l'applicazione Flask. Infatti il suo nome è "discro-be-Flask" e utilizza un altro file per fare la build. Questo container dipende dal container del database essendo che il backend deve collegarsi a MySQL per poter creare la struttura del db. Quindi questo container si avvia solo quando il container db passa il test healthcheck (viene spiegato in seguito cos'è). Questo container ha anche delle variabili di ambiente che sono la stringa di connessione al db, la chiave per criptare i token di autorizzazione e una variabile che serve a python per gestire i log.

```
environment:  
  DB_CONNECTION_STRING: mysql+pymysql://root:root@db:3306/images  
  TOKEN_ENCRYPTION: chiave_segreta_per_token  
  PYTHONUNBUFFERED: 1
```

Oltre a questo c'è anche il parametro "volumes" che serve per collegare il container con delle cartelle in locale sulla macchina host. In questo modo si può vedere il file di log che si aggiorna in tempo reale.

4.4.1.1 File build

Come detto in precedenza questo container utilizza un altro file per fare la build effettiva dell'applicazione. In questo file viene definito che il container utilizza l'immagine "python:3.10-alpine". Le immagini "alpine" sono basate su Alpine Linux e solitamente vengono utilizzate queste perché sono molto leggere e veloci. Dopodiché viene copiato il file "requirements.txt" che contiene tutte le dipendenze che necessita l'applicazione e vengono installate. Una volta che tutte le dipendenze sono presenti vengono copiati tutti i file del backend e viene fatta partire l'applicazione Flask.

4.4.2 Docker gui_web

Il servizio chiamato gui_web è il container che contiene il frontend e si chiama "discro-fe-react". Anche questo container utilizza un file separato per fare la build. Pure questo servizio ha una variabile d'ambiente ovvero l'indirizzo per comunicare con il backend.

```
environment:  
  REACT_APP_SERVER_URL: http://192.168.56.50:5000
```

Infine in questo servizio è presente il parametro "volumes" che serve per lo sviluppo in modo da poter modificare il codice e le modifiche vengono applicate in automatico anche all'interno del container senza doverlo riavviare.

4.4.2.1 File build

Anche in questo container per fare la build è presente un altro file. Questo container utilizza l'immagine "node:18-alpine" e anche in questo caso come prima cosa vengono installate le dipendenze del codice copiando i file "package.json" e "package-lock.json" e eseguendo il comando "npm ci". Infine viene copiato tutto il codice del frontend e viene eseguito.

4.4.3 Docker db

Il servizio chiamato db è il container che contiene il database. Questo container si chiama “discro-db-mysql”, utilizza l’immagine “mysql:8.0.21” e non ha ulteriori file di build esterni. Oltre a questi valori ci sono anche in questo container delle variabili d’ambiente. Queste variabili sono la password per collegarsi al database e il nome del database che crea all’avvio del container.

```
environment:  
  MYSQL_ROOT_PASSWORD: root  
  MYSQL_DATABASE: images
```

In più questo container necessita del parametro healthcheck che serve per comunicare al container del backend che il database è pronto per la connessione. Questo parametro esegue un test ogni tre secondi per un massimo di dieci tentativi e controlla se è possibile connettersi a MySQL.

```
healthcheck:  
  test: ["CMD", "mysqladmin","ping", "-h", "localhost"]  
  timeout: 3s  
  retries: 10
```

Infine c’è il parametro volumes che serve per poter rendere il database persistente salvando il database sulla macchina locale e non nel container in modo che se si ferma l’applicazione e la si fa ripartire il database conterrà ancora i valori.

5 Test

5.1 Protocollo di test

Test Case:	TC-001	Nome:	Registrazione account
Riferimento:	REQ-01	Nome:	
Descrizione:	Bisogna registrare un account al sito.		
Prerequisiti:	1. Accedere al sito web 2. Aprire la pagina registrazione		
Procedura:	1. Inserire username 2. Inserire password 3. Inserire conferma password 4. Cliccare pulsante “Sign Up”		
Risultati attesi:	L'account verrà creato con successo e si verrà portati alla pagina di login.		

Test Case:	TC-002	Nome:	Registrazione account valori non validi
Riferimento:	REQ-01	Nome:	
Descrizione:	Testare vari formati di username e password non accettati dal sito.		
Prerequisiti:	1. Accedere al sito web 2. Avere un account già esistente chiamato admin. 3. Aprire la pagina di registrazione		
Procedura:	1. Provare a registrarsi con nome utente “test Password&1” 2. Provare a registrarsi con nome utente “admin Password&1” 3. Provare a registrarsi con nome utente “testcase test” 4. Provare a registrarsi con nome utente “testcase Password1” 5. Provare a registrarsi con nome utente “testcase Password&”		
Risultati attesi:	Tutti questi tentativi dovranno ritornare un errore che verrà gestito dal sito web e mostrato all'utente.		

Test Case:	TC-003	Nome:	Login account
Riferimento:	REQ-01	Nome:	
Descrizione:	Bisogna accedere ad un account esistente.		
Prerequisiti:	1. Accedere al sito web 2. Account già esistente 3. Aprire la pagina di login		
Procedura:	1. Inserire username 2. Inserire password		
Risultati attesi:	Si caricherà l'homepage e si potrà accedere all'history dell'account. In più in alto a destra si potrà visualizzare l'username con cui si ha fatto l'accesso.		

Test Case:	TC-004	Nome:	Logout account
Riferimento:	REQ-01		
Descrizione:	Bisogna uscire dall'account sul sito.		
Prerequisiti:	1. Accedere al sito web 2. Aver fatto l'accesso con un account esistente		
Procedura:	1. Cliccare tasto logout in alto a destra.		
Risultati attesi:	L'homepage verrà ricaricata e non si vedrà più l'history e il nome utente. In più in alto a destra saranno di nuovo presenti i pulsanti "Sign in" e "Sign up"		

Test Case:	TC-005	Nome:	Generazione immagine con account
Riferimento:	REQ-06		
Descrizione:	Bisogna elaborare un'immagine applicando uno o più filtri di discromatopsia avendo effettuato l'accesso sul sito.		
Prerequisiti:	1. Accedere al sito web 2. Aver fatto l'accesso con un account esistente		
Procedura:	1. Caricare un'immagine sul sito 2. Selezionare un filtro discromatopsia 3. Cliccare "genera" e aspettare il risultato		
Risultati attesi:	Una volta che l'immagine verrà generata, questa verrà mostrata a schermo. In più l'history dovrà aggiornarsi in automatico e visualizzare l'immagine appena generata in cima.		

Test Case:	TC-006	Nome:	Generazione immagine senza account
Riferimento:	REQ-06		
Descrizione:	Bisogna elaborare un'immagine applicando uno o più filtri di discromatopsia senza aver effettuato l'accesso sul sito.		
Prerequisiti:	1. Accedere al sito web		
Procedura:	1. Caricare un'immagine sul sito 2. Selezionare un filtro discromatopsia 3. Cliccare "genera" e aspettare il risultato		
Risultati attesi:	Una volta che l'immagine verrà generata, questa verrà mostrata a schermo. In questo caso non verrà salvata da nessuna parte.		

Test Case:	TC-007	Nome:	Applicare tutti i tipi di discromatopsia
Riferimento:	REQ-06		
Descrizione:	Bisogna elaborare un'immagine applicando tutti i filtri di discromatopsia		
Prerequisiti:	1. Accedere al sito web		
Procedura:	1. Caricare un'immagine sul sito 2. Selezionare la spunta "All" 3. Cliccare "genera" e aspettare il risultato		
Risultati attesi:	Una volta cliccata la spunta "All" tutti gli altri checkbox verranno disabilitati e una volta che le immagini vengono elaborate ci saranno tutti i tipi di discromatopsia applicati alla foto originale.		

Test Case:	TC-008	Nome:	Visualizzazione history
Riferimento:	REQ-03		
Descrizione:	Bisogna visualizzare l'history delle immagini generate in precedenza		
Prerequisiti:	1. Accedere al sito web 2. Aver fatto l'accesso con un account esistente		
Procedura:	1. Scorrere fino a visualizzare l'history		
Risultati attesi:	Verranno mostrate tutte le foto generate negli ultimi 30 giorni		

Test Case:	TC-009	Nome:	Rimozione valori history
Riferimento:	REQ-04		
Descrizione:	Bisogna aprire l'history e rimuovere dei valori		
Prerequisiti:	1. Accedere al sito web 2. Aver fatto l'accesso con un account esistente		
Procedura:	1. Scorrere fino a visualizzare l'history 2. Cliccare su elimina sul valore che si vuole rimuovere		
Risultati attesi:	L'immagine verrà rimossa dal database e non si potrà più visualizzare.		

Test Case:	TC-010	Nome:	Funzionamento protocollo di logging
Riferimento:	REQ-07		
Descrizione:	Bisogna fare diverse azioni per vedere se viene loggato tutto sul server.		
Prerequisiti:	1. Accedere al sito web		
Procedura:	1. Accedere ad un account 2. Elaborare diverse immagini 3. Rimuovere immagini dall'history		
Risultati attesi:	Dal lato server queste azioni verranno loggiate in un file di testo		

Test Case:	TC-011	Nome:	Cambiamento lingua
Riferimento:	REQ-08		
Descrizione:	Bisogna cambiare lingua al sito		
Prerequisiti:	1. Accedere al sito web		
Procedura:	1. Cliccare il tasto cambia lingua 2. Selezionare un'altra lingua		
Risultati attesi:	Tutte le scritte verranno tradotte		

Test Case:	TC-012	Nome:	Test sito responsive
Riferimento:	REQ-09		
Descrizione:	Il sito dovrà adattarsi alla dimensione dello schermo		
Prerequisiti:	1. Accedere al sito web		
Procedura:	1. Cambiare dimensioni della finestra dall'ispezione		
Risultati attesi:	Il sito dovrà adattarsi e non ci dovranno essere per esempio dei componenti tagliati a metà		

Test Case:	TC-013	Nome:	Caricamento history
Riferimento:	REQ-10		
Descrizione:	La history dovrà caricarsi in modo dinamico		
Prerequisiti:	1. Accedere al sito web 2. Fare l'accesso con un account con più di 5 foto nell'history		
Procedura:	1. Scorrere fino a visualizzare l'history		
Risultati attesi:	I valori dell'"history si caricheranno man mano che si scorre la pagina di 5 in 5.		

Test Case:	TC-014	Nome:	Rimozione valori history automatico
Riferimento:	REQ-05		
Descrizione:	Controllare che i valori history si eliminano da soli dopo un tot tempo.		
Prerequisiti:	1. Accedere al sito web 2. Aver fatto l'accesso con un account esistente 3. Elaborare immagini		
Procedura:	1. Modificare valori tempistiche eliminazione 2. Controllare che immagini vengano eliminate in automatico dal db		
Risultati attesi:	L'immagine verrà rimossa in modo automatico dal database e non si potrà più visualizzare.		

Test Case:	TC-015	Nome:	Informazioni
Riferimento:	REQ-11		
Descrizione:	Aprire il pop-up che contiene le informazioni sui vari tipi di discromatopsia.		
Prerequisiti:	1. Accedere al sito web		
Procedura:	1. Cliccare sul pulsante "Information"		
Risultati attesi:	Si dovrà aprire un pop-up che mostra un esempio per ogni tipo di discromatopsia.		

5.2 Risultati test

ID	Risultato	Note	Data
TC-001	Passato	Si apre la pagina di login in modo corretto.	15.11.2023
TC-002	Passato	<p>Tutti questi tentativi di creare un account non riusciranno e verranno mostrati all'utente i seguenti errori.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <p>Error while Sign Up ×</p> <p>Username must be minimum 5 characters</p> </div> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <p>Error while Sign Up ×</p> <p>Username already taken</p> </div> <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> <p>Error while Sign Up ×</p> <p>Password must be at least 6 characters, at least one number and at least one special char</p> </div>	15.11.2023
TC-003	Passato	Facendo login con l'account creato nel TC-001 si apre l'homepage e in alto a destra si visualizza l'username.	15.11.2023
TC-004	Passato	Cliccando il tasto del logout il sito si è ricaricato e non posso più vedere l'history e l'username.	15.11.2023
TC-005	Passato	Una volta cliccato il pulsante “Generate” parte un caricamento e quando finisce verranno mostrate le immagini appena generate. Anche l'history si aggiorna aggiungendo il valore appena elaborato in cima.	17.11.2023
TC-006	Passato	Anche in questo caso le immagini vengono generate correttamente però non vengono salvate perché non si ha fatto l'accesso.	15.11.2023
TC-007	Passato	<p>Cliccando la spunta “All” tutti i checkbox si disabilitano e vengono generati tutti i tipi di discromatopsia.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p><input type="checkbox"/> Protanopia</p> <p><input type="checkbox"/> Protanomaly</p> <p><input type="checkbox"/> Deutanopia</p> <p><input type="checkbox"/> Deutanomaly</p> <p><input type="checkbox"/> Tritanopia</p> <p><input type="checkbox"/> Tritanomaly</p> <p><input type="checkbox"/> Achromatopsia</p> <p><input checked="" type="checkbox"/> All</p> </div>	15.11.2023
TC-008	Passato	Scorrendo verso il basso nella pagina web si vede l'history con i valori generati.	15.11.2023

TC-009	Passato	Quando si clicca sul pulsante “remove” le elaborazioni vengono rimosse da db e anche dal sito.	15.11.2023
TC-010	Passato	Generando delle immagini e rimuovendole questo è quello che viene scritto nel file di log. <pre>2023-11-15 09:09:35,957 - [INFO] - 192.168.56.1 - - [15/Nov/2023 09:09:35] "OPTIONS /generate HTTP/1.1" 200 - 2023-11-15 09:09:35,999 - [INFO] - new album id 3 created by admin 2023-11-15 09:09:36,033 - [INFO] - image successfully uploaded on database in album 3 with type original 2023-11-15 09:09:36,033 - [INFO] - image to process for user admin from 192.168.56.1 2023-11-15 09:09:42,170 - [INFO] - 192.168.56.1 - - [15/Nov/2023 09:09:42] "OPTIONS /remove HTTP/1.1" 200 - 2023-11-15 09:09:42,175 - [INFO] - album to remove: 3 2023-11-15 09:09:42,205 - [INFO] - album successfully removed</pre>	15.11.2023
TC-011	Passato	Cliccando sulla bandiera in alto a sinistra e selezionando la seconda lingua tutte le scritte verranno cambiate nella lingua corretta.	15.11.2023
TC-012	Passato	Impostando come dimensione dello schermo quella di un telefono l’interfaccia si adatta e cambia la disposizione di certi componenti in modo da poter visualizzare tutto senza alcun problema.	15.11.2023
TC-013	Passato	Mentre si scorre la pagina l’history le elaborazioni vengono caricate di 5 in 5.	15.11.2023
TC-014	Non passato	La scheduled task per rimuovere i valori vecchi dal container Docker non parte.	15.11.2023
TC-015	Passato	Il pop-up si apre correttamente mostrando degli esempi all’utente.	15.11.2023

5.3 Test REST API

Oltre ad aver effettuato i test case ho anche creato dei test automatizzati con Postman per testare la REST API. Infatti nella cartella 7_Allegati sotto la cartella Test è presente il file Postman_Test_RESTapi.json che si può importare su Postman e farli partire tutti assieme.

```

POST Register
POST Register_Same_User
POST Register_Invalid_Password
POST Register_Short_Username
POST Register_Long_Username
GET Login
GET Login_Token
GET Login_Failed
POST Generate
GET History
GET History_Failed
DEL Remove
DEL Remove_Failed
DEL Remove_Old

```

Figura 28 Test automatizzati

Questi sono tutte le richieste che vengono eseguite come test automatizzati. Quasi tutte le richieste hanno 2 test che devono superare ovvero un controllo che viene ritornato lo status code corretto (che viene controllato su tutte le richieste) e un controllo di altri valori nella risposta (non tutte le richieste hanno questo controllo).

```
pm.test("request status not authorized", function() {  
    pm.response.to.have.status(401)  
})
```

Questo, per esempio, è il test della richiesta “Login_Failed” e infatti come test viene controllato che la risposta abbia come status code 401 che significa “Autenticazione non valida”. In più certe richieste si salvano nelle variabili di Postman dei valori che vengono utilizzati in seguito. Per esempio quando si fa la richiesta del Login valida viene salvato il token in modo da autenticarsi con quello nelle richieste successive.

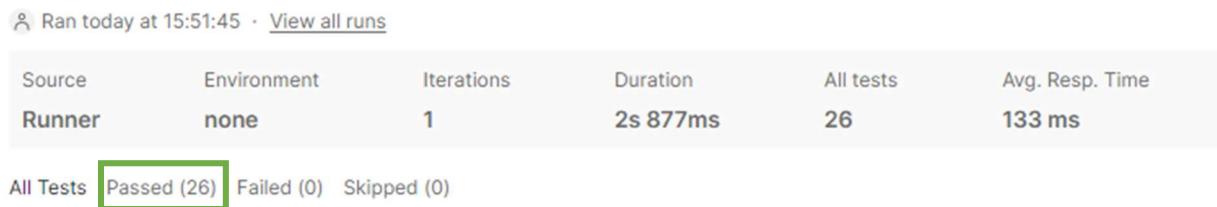


Figura 29 Risultato test Postman

Questo è il risultato che ottengo facendo partire tutti i test ovvero su 26 test tutti e 26 passano. Nella cartella allegati si trova anche un JSON contenente i risultati che si può anche questo importare su Postman.

5.4 Mancanze/limitazioni conosciute

Purtroppo, non sono riuscito a capire il motivo, ma la scheduled Task non funziona all'interno del container Docker. Infatti ho dovuto creare una route come workaround in modo che la funzionalità si possa utilizzare anche se non parte in automatico.

6 Consuntivo

Qua sotto si può vedere il Gantt consuntivo con le differenze rispetto al Gantt preventivo.

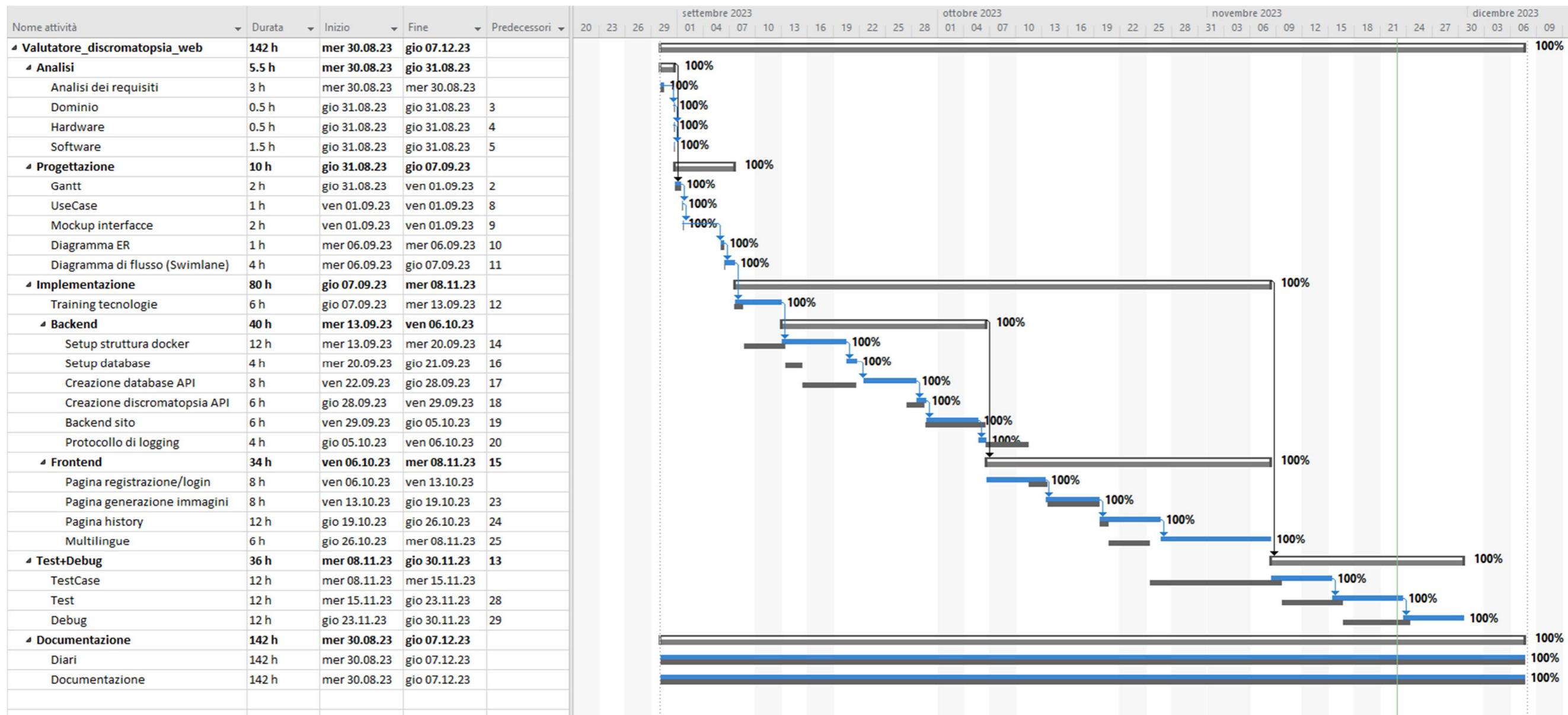


Figura 30 Gantt consuntivo

6.1 Differenze con Gantt preventivo

Come si vede dal diagramma di Gantt consuntivo sovrastante ci sono state delle differenze rispetto a quello che avevo pianificato:

- La differenza più evidente è sicuramente che manca l'attività “Creazione Storage API” che ho rimosso perché ho voluto cambiare la mia idea iniziale di come sviluppare l'applicazione: infatti all'inizio l'idea era di salvare le immagini all'esterno del database e di salvarmi la path per raggiungere l'immagine nel db, però alla fine ho optato per salvare le immagini direttamente nel database in dei Longblob perché mi trovavo più comodo con questo metodo. Avendo rimosso quest'attività sono riuscito a guadagnare un po' di tempo che ho sfruttato per implementare al meglio altre funzionalità.
- Un'altra differenza molto grande è la tempistica con il quale avevo previsto di creare la struttura con Docker. Infatti nel Gantt preventivo avevo previsto di metterci quattro ore ma alla fine ci ho messo tre volte tanto. Questa grande differenza è stata causata dal fatto che non avevo alcuna conoscenza di Docker e anche perché non ho utilizzato il metodo corretto per lavorarci.
- Un'ulteriore differenza abbastanza evidente è la tempistica dell'implementazione dell'history, infatti ho impiegato il doppio di quanto avevo previsto. Questo è stato causato dal fatto che ho aggiunto una funzionalità non prevista ovvero l'history che si carica in modo dinamico (non richiesta nel Qdc e non ancora prevista nel Gantt preventivo) che, giustamente, mi ha dato più lavoro del previsto.
- Però, in generale, sono riuscito comunque a gestire bene i tempi perché avevo già previsto due settimane di margine che dovevo dedicare solo alla documentazione, in più, come ho detto sopra, un'attività (dalla durata di 8 ore) non l'ho svolta perché ho cambiato idea su come sviluppare l'applicazione e anche ad altre attività dove sono riuscito a guadagnarci un po' di ore.

7 Conclusioni

Arrivato alla fine del progetto posso dire che del mio lavoro funziona tutto tranne una funzionalità, ovvero la rimozione dei dati vecchi. Questo prodotto si può utilizzare veramente e può tornare utile a chiunque voglia fare un rapido test per la discromatopsia.

7.1 Sviluppi futuri

Per questo progetto si potrebbero sviluppare delle migliorie come:

- Rendere la grafica del sito più coerente essendo che non tutte le pagine hanno lo stesso stile.
- Riuscire a far funzionare la scheduled task per l'eliminazione automatica dei valori dal database all'interno del container Docker.
- Rendere il backend più efficiente implementando l'elaborazione delle immagini multithread.

In più a questo progetto si potrebbero aggiungere delle estensioni come:

- Elaborazione di video o gif
- Scattare la foto da mobile ed elaborarla subito
- Aggiungere dei filtri sulla data nell'history
- Aggiungere la funzionalità di salvare degli album nei preferiti per averli sempre in cima all'history oppure per non venir eliminati automaticamente.
- Aggiungere un pulsante che scarica le foto elaborate
- Aggiungere la condivisione delle proprie elaborazioni

7.2 Considerazioni personali

Grazie a questo progetto ho imparato ad utilizzare Python e Docker, due tecnologie che non avevo mai utilizzato prima d'ora. In più ho anche migliorato le mie conoscenze con React. Come funzionalità ho imparato a sviluppare una REST API e ad utilizzare un ORM. In generale posso dire che svolgere questo progetto mi è piaciuto perché, oltre ad aver appreso nuove conoscenze, sono contento del risultato che ho ottenuto. La parte che ho fatto più fatica a sviluppare è stata la creazione della struttura del Docker perché era qualcosa che non avevo mai visto, ma alla fine sono riuscito a cavarmela comunque. Con le tecnologie che ho scelto mi sono trovato molto bene e anche a sviluppare un'applicazione separando frontend e backend mi è piaciuto.

8 Glossario

Termine	Descrizione
Backend	Il backend è la parte dell'applicazione che non si interfaccia con l'utente, quindi dove i dati vengono elaborati e, nel caso, salvati nel database.
Decoratore di python	Un decoratore in python serve per aggiungere delle funzionalità a delle funzioni senza dover modificare la struttura. Nel mio codice vengono utilizzati per collegare le funzioni create da me alle routes della REST API e anche per controllare l'autenticazione della richiesta prima di eseguire le funzioni.
Frontend	Il frontend è la parte dell'applicazione che si interfaccia con l'utente, quindi la GUI e la raccolta dei dati che poi vengono inviati al backend.
JSON	Javascript Object Notation: è un formato di rappresentazione dei dati basato sul linguaggio Javascript.
Listener	Un listener in un sito web serve per rilevare su l'utente esegue certe azioni per esempio se l'utente clicca da qualche parte sul sito, cambia dimensione della finestra, scorre sulla pagina, etc.
Longblob	Il longblob è un tipo di attributo di MySQL che permette di salvare dei file binari fino ad una lunghezza massima di 4,294,967,295 bytes.
ORM	Object Relational Mapping: è una tecnica per poter creare utilizzare un database tramite la programmazione ad oggetti. Ogni tabella del database è una classe all'interno del codice.
React hook	Gli hook di React sono delle funzioni che permettono di gestire lo stato e il ciclo della vita dei componenti.
Regular expression	Una regular expression serve per vedere se una stringa soddisfi delle condizioni (per esempio contenere almeno un carattere speciale).
REST API	Representational State Transfer API: è uno stile di architettura di API basata su http dove per ogni richiesta bisogna inviare l'autenticazione (con username e password o con un token identificativo) e viene ritornata la risorsa richiesta in JSON o in XML.
Scheduled Task	Una Scheduled Task è una funzione che viene eseguita ogni intervallo di tempo previsto.
SHA-256	Lo SHA-256 è un algoritmo di hash che ritorna una stringa di 64 caratteri.
Token	Un token è una stringa di caratteri che viene utilizzata per autenticarsi senza dover utilizzare ad ogni richiesta le proprie credenziali.
Workaround	Con workaround si intende di ottenere una soluzione temporanea per raggirare un problema che non si riesce a risolvere.

9 Sitografia

1. <https://flask.palletsprojects.com/en/3.0.x/>, Documentazione Flask, 07.09.2023
2. <https://ant.design/docs/react/introduce>, Documentazione Antd, 07.09.2023
3. <https://docs.docker.com/>, Documentazione Docker, 07.09.2023
4. <https://flask-sqlalchemy.palletsprojects.com/en/3.1.x/>, Documentazione SQLAlchemy, 20.09.2023
5. <https://www.digitalocean.com/community/tutorials/how-to-use-flask-sqlalchemy-to-interact-with-databases-in-a-flask-application>, Guida utilizzo SQLAlchemy con Flask, 20.09.2023
6. <https://pillow.readthedocs.io/en/stable/reference/Image.html>, Documentazione PIL_Image python, 28.09.2023
7. <https://blog.miquelgrinberg.com/post/restful-authentication-with-flask>, Guida Autenticazione REST API, 29.09.2023
8. <https://regex101.com/>, Test regular expression, 29.09.2023
9. <https://apscheduler.readthedocs.io/en/3.x/userguide.html>, Documentazione scheduled task, 04.10.2023
10. <https://axios-http.com/docs/intro>, Documentazione axios, 07.10.2023
11. <https://uiverse.io/loaders>, Loaders open source fatti con CSS, 18.10.2023
12. <https://react.i18next.com/>, Documentazione i18n, 26.10.2023
13. <https://learning.postman.com/docs/writing-scripts/test-scripts/>, Documentazione test Postman, 21.11.2023

10 Allegati

QdC	1_QdC/ QdC_1Sem23-24_M_PeGe_Valutatore_discromatopsia_web.docx
Abstract	2_Abstract/Abstract.docx
Diari di lavori	4_Diari/
Schemi Database	6_Database/Schemi/
Diagrammi di Gantt	7_Allegati/Gantt/
Diagrammi Swimlane	7_Allegati/Swimlane/
Test automatizzati	7_Allegati/Test/
Mockup	7_Allegati/Mockup/
Schema di rete	7_Allegati/SchemaRete/
Use Case	7_Allegati/UseCase