

3D Face Modelling

Titolo del progetto	3D Face Modelling
Alunno/a	Luca Lorenzon
Classe	I4AA
Anno scolastico	2023/2024
Docente responsabile	Geo Petrini

1 Indice

1	Indice.....	2
2	Introduzione.....	4
2.1	Informazioni sul progetto.....	4
2.2	Abstract.....	4
2.3	Scopo.....	4
3	Analisi.....	5
3.1	Analisi del dominio.....	5
3.2	Analisi e specifica dei requisiti.....	5
3.3	Use case.....	8
3.3.1	Utente.....	8
3.3.2	Server.....	8
3.4	Pianificazione.....	9
3.4.1	Milestone.....	11
3.5	Analisi dei mezzi.....	12
3.5.1	Software.....	12
3.5.2	Librerie principali.....	12
3.5.3	Hardware.....	13
4	Progettazione.....	14
4.1	Design dell'architettura del sistema.....	14
4.2	Design dei dati e database.....	15
4.2.1	User.....	15
4.2.2	Elaboration.....	15
4.3	Design delle interfacce.....	16
4.4	Design procedurale.....	21
4.4.1	Swimlane elaborazione immagini.....	22
5	Implementazione.....	24
5.1	Backend.....	24
5.1.1	Struttura Cartelle.....	24
5.1.2	Database.....	25
5.1.3	REST API.....	27
5.1.4	Socket.IO.....	30
5.1.5	Elaborazione modelli.....	31
5.1.6	Gestione errori.....	34
5.2	Frontend.....	35
5.2.1	Struttura Cartelle.....	35
5.2.2	Pagine e navigazione.....	36
5.2.3	Comunicazione con backend.....	44
5.2.4	Certificati.....	46
6	Test.....	47
6.1	Protocollo di test.....	47
6.2	Risultati test.....	53
6.2.1	Errore non conosciuto.....	58
6.3	Mancanze/limitazioni conosciute.....	58

7	Consuntivo	59
7.1	Differenze con il Gantt preventivo	60
8	Conclusioni	61
8.1	Sviluppi futuri.....	61
8.2	Considerazioni personali.....	61
9	Bibliografia.....	62
9.1	Sitografia	62
10	Glossario.....	63
11	Indice delle figure	64
12	Allegati	65

2 Introduzione

2.1 Informazioni sul progetto

Informazioni generali:

- Sezione: Informatica
- Nome Progetto: 3D Face Modelling
- Allievo: Luca Lorenzon
- Classe: I4AA
- Docente Responsabile: Geo Petrini
- Data Inizio: 14.12.2023
- Data Fine: 28.03.2024
- Link Repository: http://gitsam.cpt.local/2023_2024_2_semestre/3d-face-modelling

2.2 Abstract

The goal of this project is to create a product that allows the user to create a 3D model of their face. To achieve this goal, I will create a website where a user can take a picture of themselves or upload a photo. This image will be used by an algorithm to create the 3D model. The algorithm for the elaboration uses a Machine Learning model already existent that calculates several values that are used to create the 3D model. Inside the website it is possible to log in to save your elaboration result inside of an history log. The models are visible inside the website from all the directions and you can also download it. In this document it is described the development of the whole project.

2.3 Scopo

Lo scopo del progetto è quello di creare un sito web che permette di creare dei modelli 3D passandogli una foto della propria faccia. Il backend del sito si occuperà di ritagliare la faccia del soggetto della foto ed elaborarla per poter creare un modello 3D della faccia. Una volta finita l'elaborazione il modello 3D verrà mostrato all'utente che avrà anche la possibilità di scaricarlo. Sul sito sarà anche presente l'history per poter accedere alle proprie elaborazioni vecchie.

3 Analisi

3.1 Analisi del dominio

Questo progetto consiste in un sito web che crea dei modelli 3D partendo da un'immagine passata dall'utente. Sul sito sarà anche possibile registrarsi per poter salvare le proprie elaborazioni all'interno dell'history. Per sviluppare questo progetto ho diviso il backend dal frontend. Per il backend utilizzerò Python mentre per il frontend andrò ad utilizzare il framework Sveltekit. Questo sito sarà indirizzato a chiunque sia interessato ad un modello 3D della propria faccia. Ad oggi non ho trovato un prodotto simile online.

3.2 Analisi e specifica dei requisiti

Req-001	
Nome	Pagina di Registrazione
Priorità	1
Versione	1.0
Note	Sul sito sarà possibile registrarsi con un username e una password

Req-002	
Nome	Pagina di Login
Priorità	1
Versione	1.0
Note	Sul sito sarà possibile fare l'accesso con un username e una password

Req-003	
Nome	Pagina Principale
Priorità	1
Versione	1.0
Note	La pagina principale permetterà di caricare l'immagine da elaborare.

Req-004	
Nome	Upload immagine
Priorità	1
Versione	1.0
Note	Sarà possibile caricare un'immagine oppure catturarla tramite la webcam per poi elaborarla.

Req-005	
Nome	History
Priorità	1
Versione	1.0
Note	Nella pagina principale è presente l'history contenente le vecchie elaborazioni (sarà disponibile solo per gli utenti registrati).

Req-006	
Nome	Elaborazione immagini
Priorità	1
Versione	1.0
Note	Dall'immagine dell'utente verrà estrapolata la faccia e da qui verrà creato un modello 3D
Sotto requisiti	
Req-006_1	I modelli dovranno essere preferibilmente in formato OBJ o STL
Req-006_2	Sul sito sarà visualizzabile il progresso dell'elaborazione

Req-007	
Nome	Visualizzazione modello 3D
Priorità	1
Versione	1.0
Note	Sul sito sarà possibile visualizzare il modello 3D appena generato con la possibilità di ruotarlo in tutte le direzioni
Sotto requisiti	
Req-006_1	Sarà possibile anche scaricare i modelli 3D

Req-008	
Nome	Applicazione texture
Priorità	3
Versione	1.0
Note	Ai modelli 3D verrà applicata una texture estrapolata dall'immagine

Req-009	
Nome	Sistema di log
Priorità	1
Versione	1.0
Note	Il backend dovrà avere un sistema di logging

Req-010	
Nome	Gestione coda elaborazioni
Priorità	2
Versione	1.0
Note	Il sistema di elaborazione dovrà avere una coda per gestire al meglio le elaborazioni.

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.

3.3 Use case

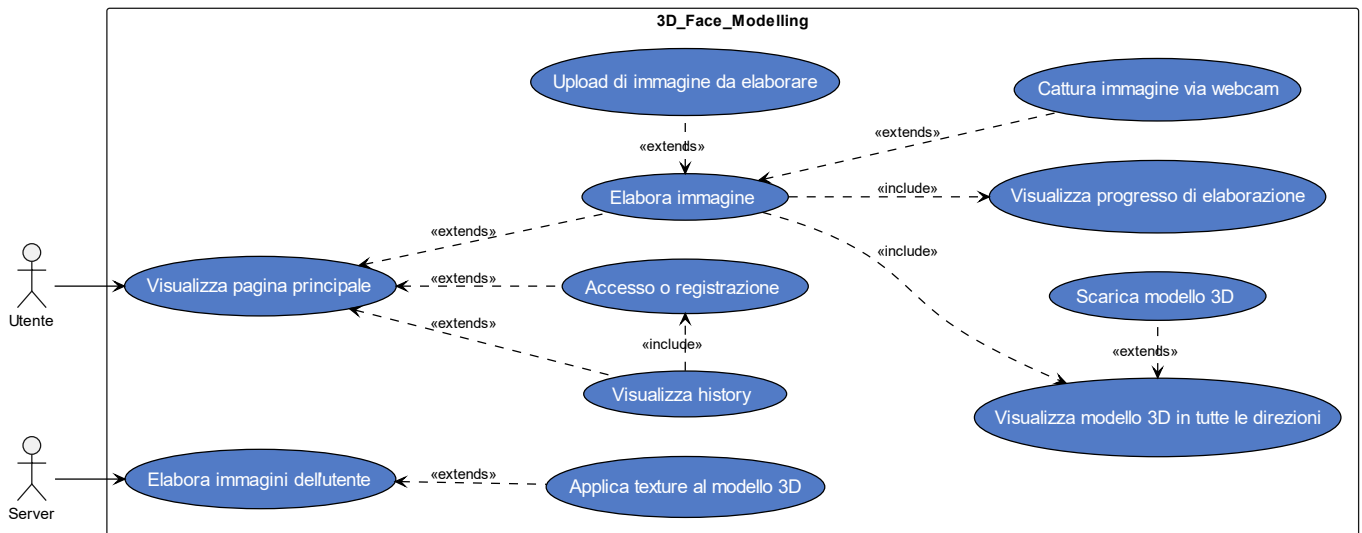


Figura 1 Use Case

In questo progetto sono presenti 2 attori: l'utente e il server.

3.3.1 Utente

L'utente all'interno del sito potrà visualizzare la pagina principale senza dover fare per forza l'accesso. Nella pagina principale è presente anche l'history che necessita però di aver effettuato l'accesso per visualizzarla. Nella pagina principale si possono elaborare le immagini per creare dei modelli 3D. Per farlo si possono caricare delle immagini sul sito oppure catturare un'immagine sul momento tramite la webcam. Mentre l'immagine viene elaborata dal backend, sul sito si potrà visualizzare il progresso dell'elaborazione. Una volta finita l'elaborazione si potrà visualizzare il modello 3D sul sito e si potrà ruotare in tutte le direzioni e ci sarà anche la possibilità di scaricarlo.

3.3.2 Server

Il server invece avrà il compito di elaborare le immagini caricate dall'utente e, opzionalmente, applicarci anche una texture estrapolata dall'immagine al modello 3D.

3.4 Pianificazione

Per questo progetto ho deciso di utilizzare il metodo kanban.

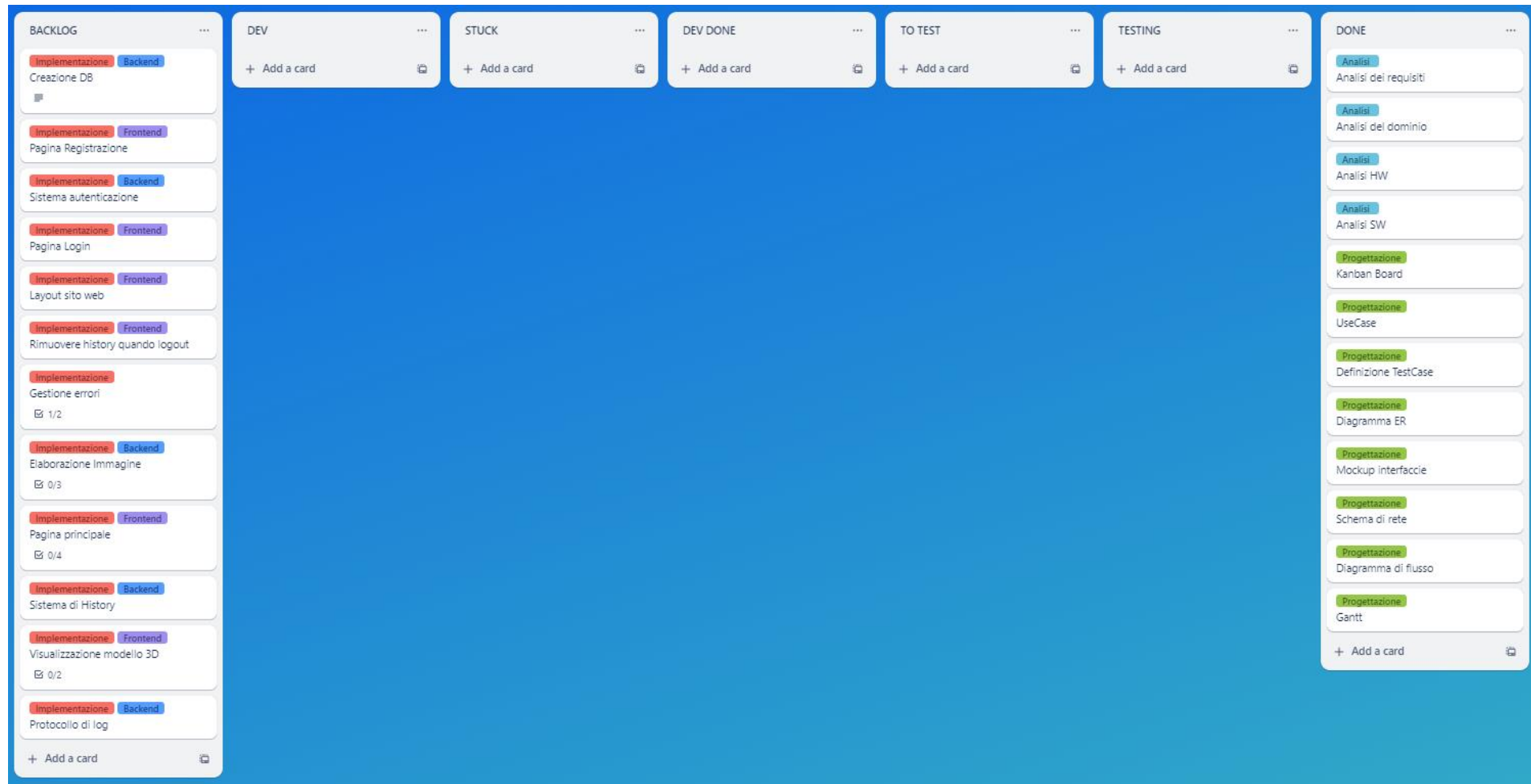


Figura 2 Kanban Board

Inoltre ho dovuto comunque creare un diagramma di Gantt dove ho anche dichiarato delle Milestone.

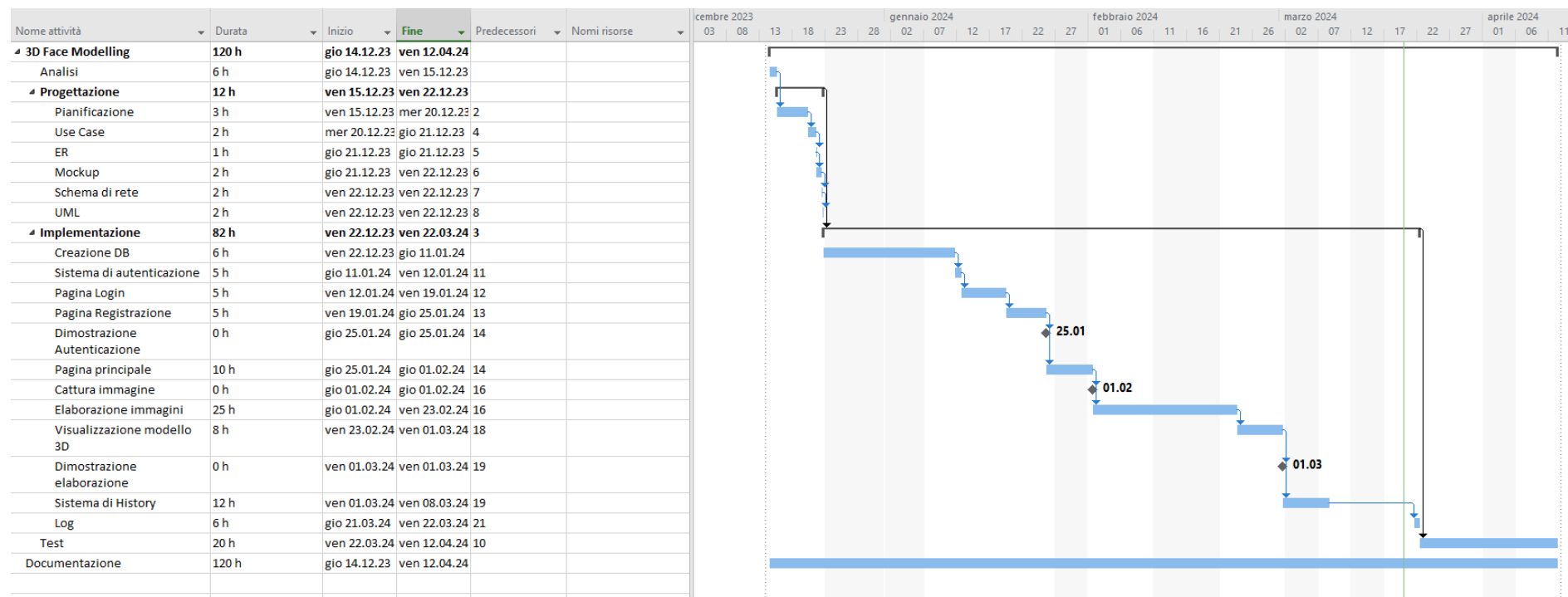


Figura 3 Diagramma di Gantt

3.4.1 Milestone

3.4.1.1 Dimostrazione Autenticazione

Questa Milestone è da presentare al 26.01.2024 e bisogna mostrare il sistema d'autenticazione funzionante seguendo i seguenti punti:

- Creazione di un account
- Accesso ad un account esistente
- Logout dall'account

3.4.1.2 Cattura Immagine

Questa Milestone è da presentare al 02.02.2024 e bisogna mostrare la cattura di un'immagine tramite la webcam e i seguenti punti:

- Cattura immagine con webcam
- Upload immagine
- Invio immagine al backend
- Dimostrazione della coda

3.4.1.3 Dimostrazione Elaborazione

Questa Milestone è da presentare al 07.03.2024 e bisogna mostrare il funzionamento dell'elaborazione delle immagini da parte del backend e della visualizzazione del risultato sul sito web. In particolare, i punti da dimostrare sono:

- Visualizzazione del progresso dell'elaborazione sul sito
- Visualizzazione risultato
- Download del risultato

3.5 Analisi dei mezzi

3.5.1 Software

Per lo sviluppo del progetto ho utilizzato i seguenti software:

- Visual Studio Code 1.78.2
- PlantUML VSCode Extension 2.17.5
- Remote – SSH VSCode Extension 0.102.0
- Microsoft Word 2019
- Microsoft Project 2019
- Microsoft Visio 2019
- Luna Modeler 7.5.1
- Google Chrome
- Firefox
- CUDA Toolkit 11.6
- Adb 28.0.2-debian

In più ho anche utilizzato i seguenti siti web:

- <https://trello.com>
- <https://figma.com>
- <https://www.lucidchart.com/>

3.5.2 Librerie principali

3.5.2.1 Backend

- Pytorch 1.13.0
- Pytorch3d 0.7.5
- Flask 3.0.2
- Socket.IO 5.3.6
- Opencv 4.9.0.80
- Pillow 10.2.0

3.5.2.2 Frontend

- Svelte 4.2.8
- Tailwind 3.4.1
- Skeleton 2.7.0
- Vite 5.0.12
- Threlte 7.2.1

3.5.3 Hardware

Per lo sviluppo di questo progetto mi è stato fornito dalla scuola un PC 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

Mi è stato fornito anche un secondo PC verso la metà del progetto perché necessitavo di lavorare in un ambiente Linux per scaricare certe dipendenze. Questo PC ha le seguenti caratteristiche:

- Sistema Operativo: Linux Mint 21.3.
- Processore: Intel(R) Core (TM) i3-4150 CPU @ 3.50GHz
- RAM: 8.0 GB
- GPU: Nvidia GTX 970

In più mi è stata fornita dalla scuola una webcam per poter fare la cattura di immagini sul sito web.

4 Progettazione

4.1 Design dell'architettura del sistema

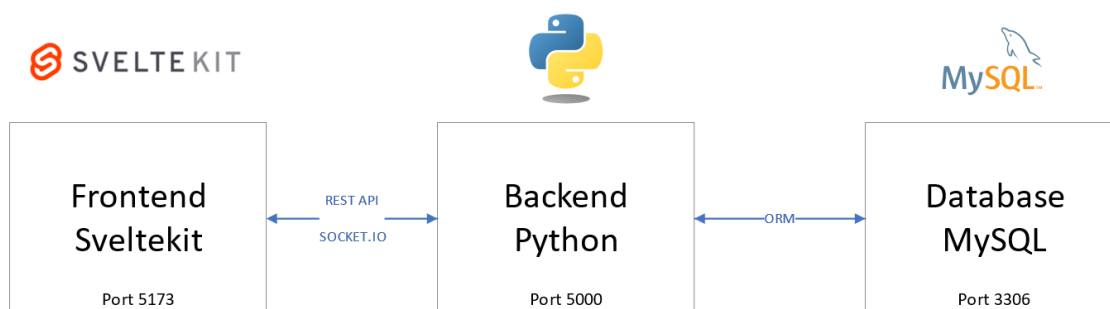


Figura 4 Schema di rete

Questo schema mostra come sarà strutturata l'applicazione. Infatti l'applicazione è suddivisa in 3 parti il frontend, il backend e il database. Il frontend sarà sviluppato in Typescript con il framework Sveltekit e l'utente potrà raggiungerlo sulla porta 5173. Il backend sarà sviluppato in Python con il micro-framework Flask e sarà raggiungibile dal frontend sulla porta 5000. Il frontend e il backend comunicheranno tramite una REST API e Socket.IO. Infine sarà presente il database in MySQL al quale si conatterà il backend sulla porta 3306 ed eseguirà le azioni tramite un ORM.

4.2 Design dei dati e database

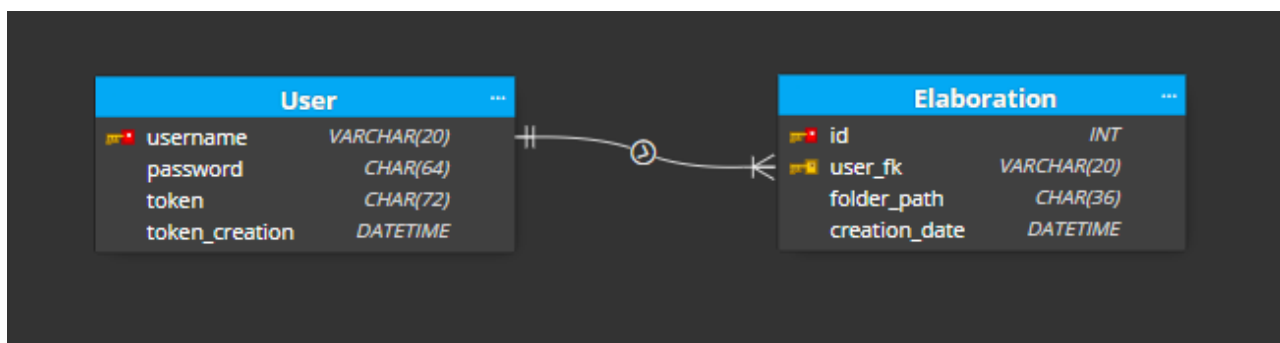


Figura 5 Schema logico

Il database di quest'applicazione è molto semplice, infatti è composto solo da due tabelle: la tabella User e la tabella Elaboration.

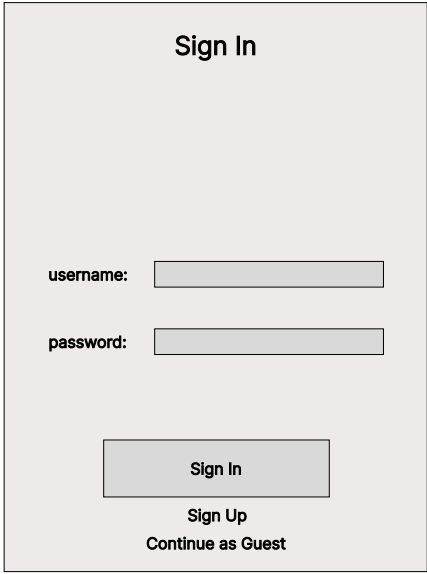
4.2.1 User

La tabella User conterrà solamente lo username, la password di un account e il token dell'account per autenticarsi con la data di creazione. L'username sarà la chiave primaria e potrà essere lungo un massimo di 20 caratteri. La password invece sarà sempre lunga 64 caratteri perché viene salvata nel database dopo che verrà applicato l'algoritmo di hash SHA-256 che ritorna sempre una stringa da 64 caratteri. Il campo token ha una lunghezza fissa di 72 caratteri perché viene utilizzato un metodo di salting e di hash per creare i token. Infine il campo della data di creazione del token serve per controllare quando scade il token e quindi bisogna eseguire di nuovo il login.

4.2.2 Elaboration

La tabella Elaboration invece conterrà un id, un utente, il nome di una cartella e la data di creazione. L'id sarà la chiave primaria in modo da poter identificare le varie elaborazioni. L'attributo user_fk sarà una foreign key alla tabella User in modo da poter definire a chi appartiene l'elaborazione. Il nome di una cartella serve per definire dove verranno salvati i due file che compongono le elaborazioni (l'immagine e il modello 3D). Questo campo ha una lunghezza fissa di 36 caratteri perché le cartelle delle elaborazioni sono chiamate con degli uuid in modo da essere univoci. Infine la data serve per poter ordinare l'history in ordine cronologico.

4.3 Design delle interfacce



Sign In

username:

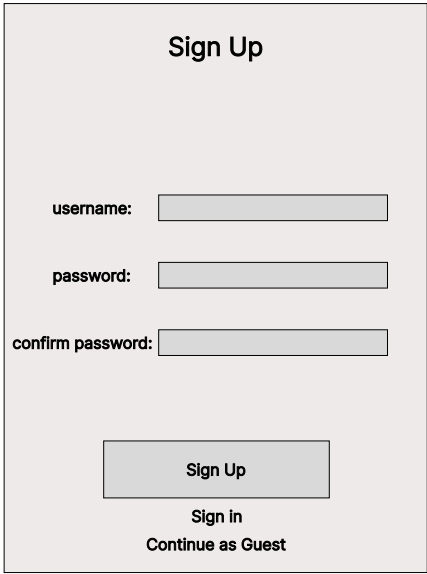
password:

Sign In

Sign Up
Continue as Guest

Figura 6 Pagina di login

Nella pagina di login si può accedere al proprio account sul sito per poter vedere i propri modelli elaborati nell'history e per poter salvare le prossime elaborazioni. Per fare l'accesso è richiesto un nome utente e una password. Da questa pagina si può accedere alla pagina di registrazione in modo da poter creare un account nel caso non lo si abbia oppure si può anche tornare alla pagina principale senza fare l'accesso essendo che non è necessario per poter generare i modelli.



Sign Up

username:

password:

confirm password:

Sign Up

[Sign in](#)

[Continue as Guest](#)

Figura 7 Pagina di registrazione

Nella pagina di registrazione si può creare un account per poter salvare le proprie elaborazioni. Per poter creare un account bisogna inserire un nome utente e una password due volte per poterla confermare. In questa pagina si può anche accedere alla pagina di login per poter accedere nel caso si avesse già un account esistente oppure anche qua si può tornare alla pagina principale per poter fare delle elaborazioni senza eseguire l'accesso.

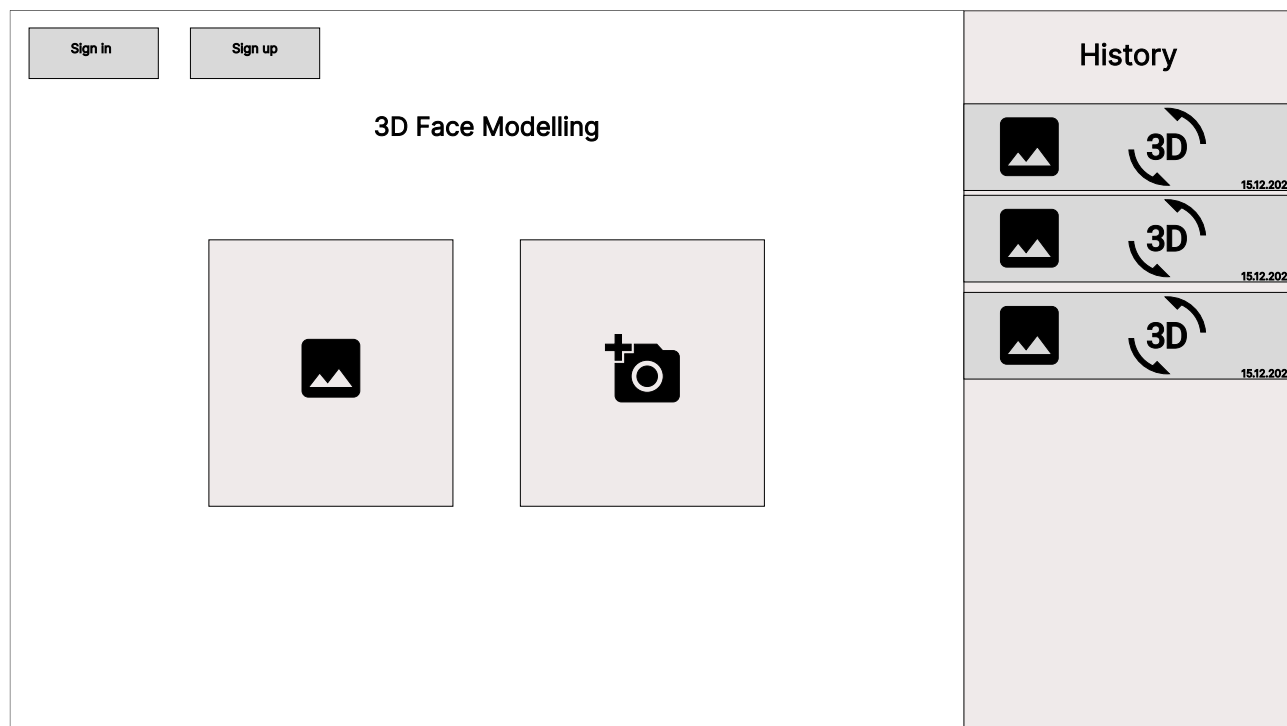


Figura 8 Pagina Principale

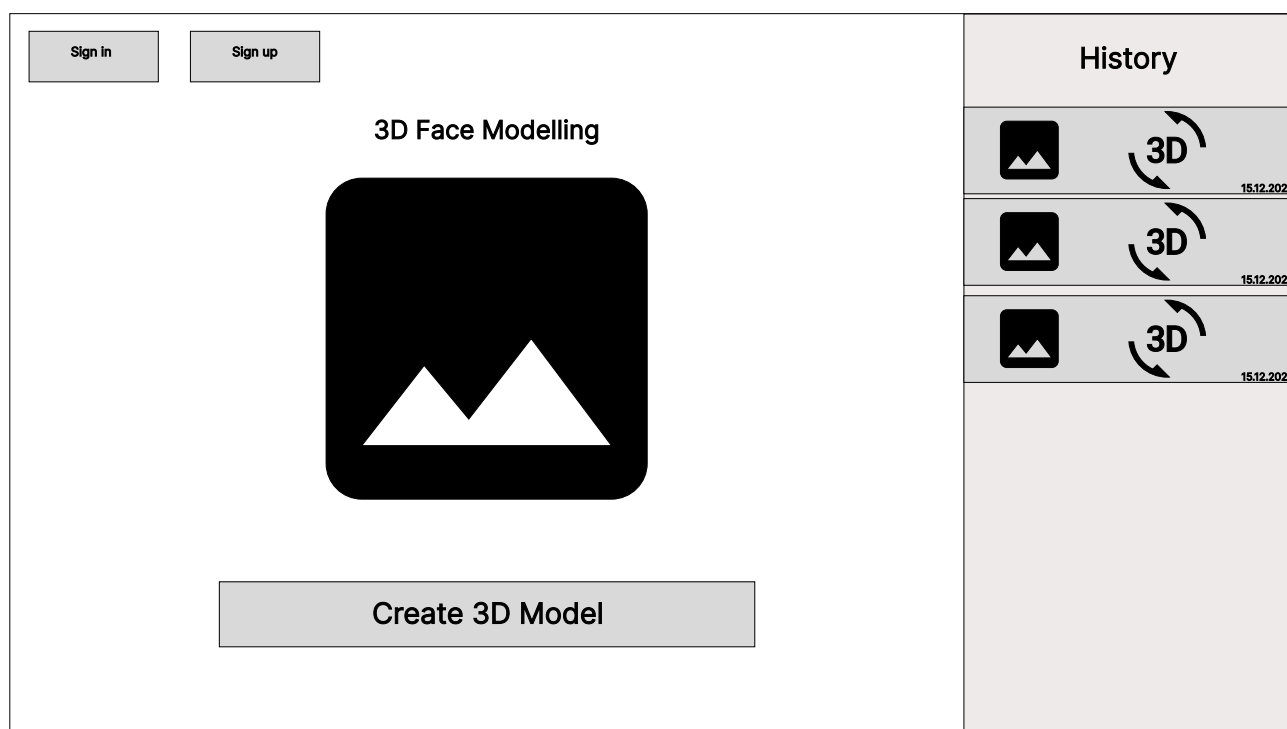


Figura 9 Pagina Principale con immagine caricata

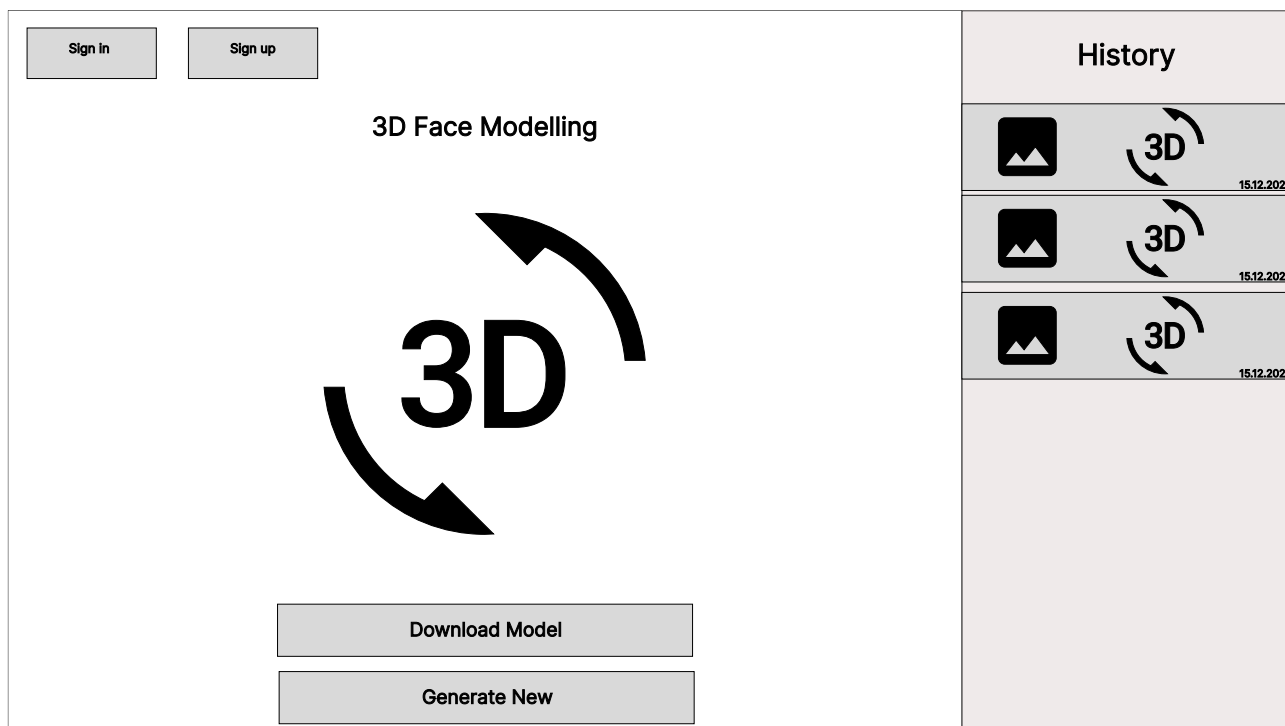


Figura 10 Pagina Principale con elaborazione

Nella pagina principale è dove si potrà inizializzare un'elaborazione scegliendo se caricare una foto oppure se scattarne una sul momento con la webcam. Una volta selezionata l'immagine verrà mostrata a schermo e verrà chiesta conferma per creare il modello 3D con l'immagine. Nel mentre che l'immagine viene elaborata verrà mostrata una barra di progressione al centro dello schermo. Una volta finita l'elaborazione verrà mostrato il risultato visualizzabile in tutte le direzioni, con la possibilità di scaricarlo e di fare una nuova elaborazione. A destra della pagina principale è presente l'history (se si ha eseguito l'accesso). In alto a sinistra invece sono presenti i tasti per registrarsi o per fare login. Quando si esegue l'accesso al posto di questi tasti sarà visualizzato il nome utente e un tasto per fare logout.

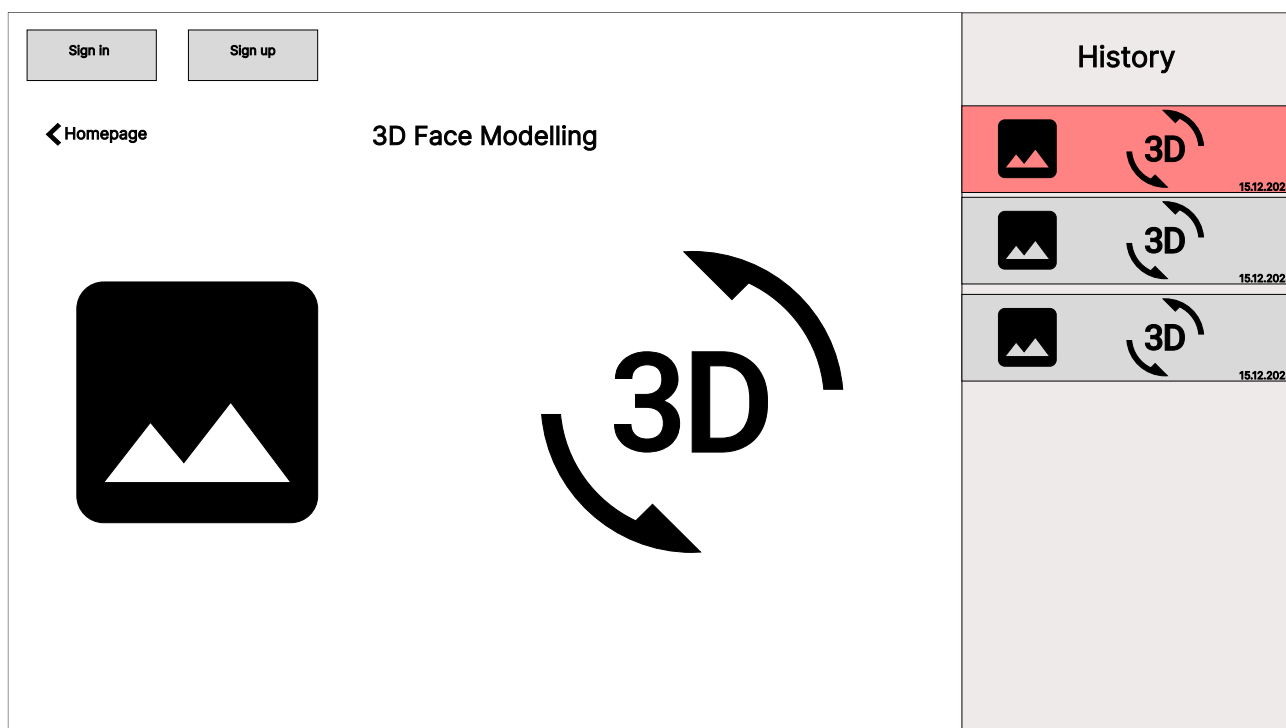


Figura 11 Elemento dell'history

Quando si clicca su un valore dell'history si aprirà una pagina che mostra l'immagine originale e il modello 3D elaborato. A sinistra sarà presente il tasto per tornare alla pagina principale e a destra sarà sempre presente la lista dell'history in modo da poter cambiare l'elemento da visualizzare. Nell'history al lato si vedrà anche quale elemento si sta visualizzando in quel momento.

4.4 Design procedurale

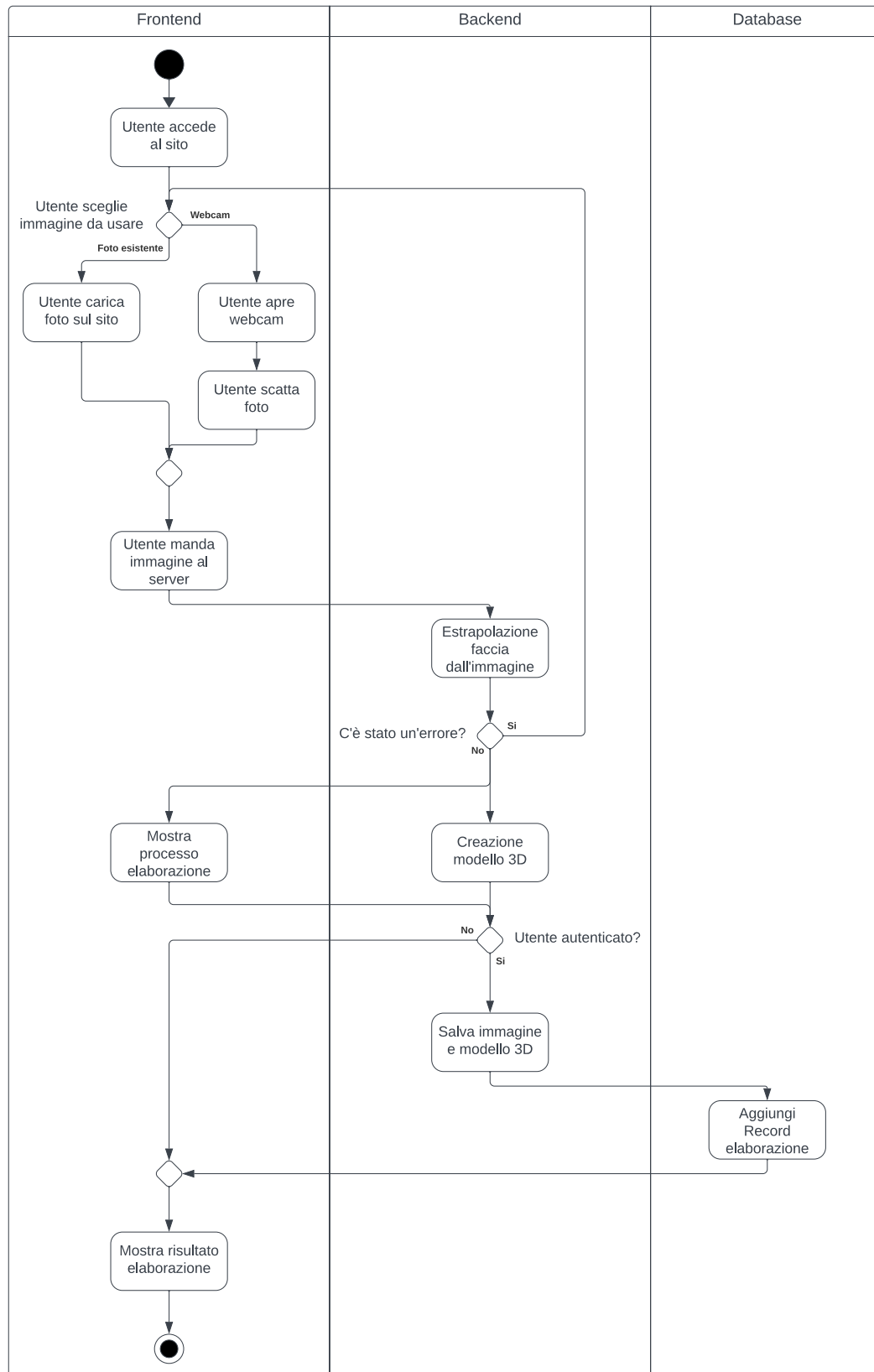


Figura 12 UML elaborazione

4.4.1 Swimlane elaborazione immagini

Questo schema mostra il funzionamento dell'elaborazione delle immagini. Il risultato finale che si vuole ottenere è quello di visualizzare sul browser il modello 3D ottenuto e poterlo ruotare in tutte le direzioni.

Per farlo come prima cosa si controlla se l'utente vuole utilizzare una sua foto già esistente oppure se ne vuole scattare una in quel momento aprendo la webcam. Una volta caricata l'immagine sul sito l'utente manda l'immagine al server dove verrà elaborata. Per farlo viene prima di tutto ritagliata la faccia dall'immagine. Se quest'azione scatena un errore il server lo ritorna all'utente che deve così ricaricare un'immagine. Se invece l'operazione va a buon fine il server comincia a creare il modello 3D della faccia. Nel mentre che il modello 3D viene creato sul sito viene mostrato il processo dell'elaborazione.

Una volta finita l'elaborazione, se si ha eseguito l'accesso, l'immagine iniziale e il modello 3D vengono salvati sul server e viene aggiunto un record nel database che far riferimento all'elaborazione appena eseguita dopo viene mostrata l'elaborazione all'utente sul sito. Se invece non si ha fatto l'accesso sul sito viene mostrato il risultato direttamente senza venir salvato.

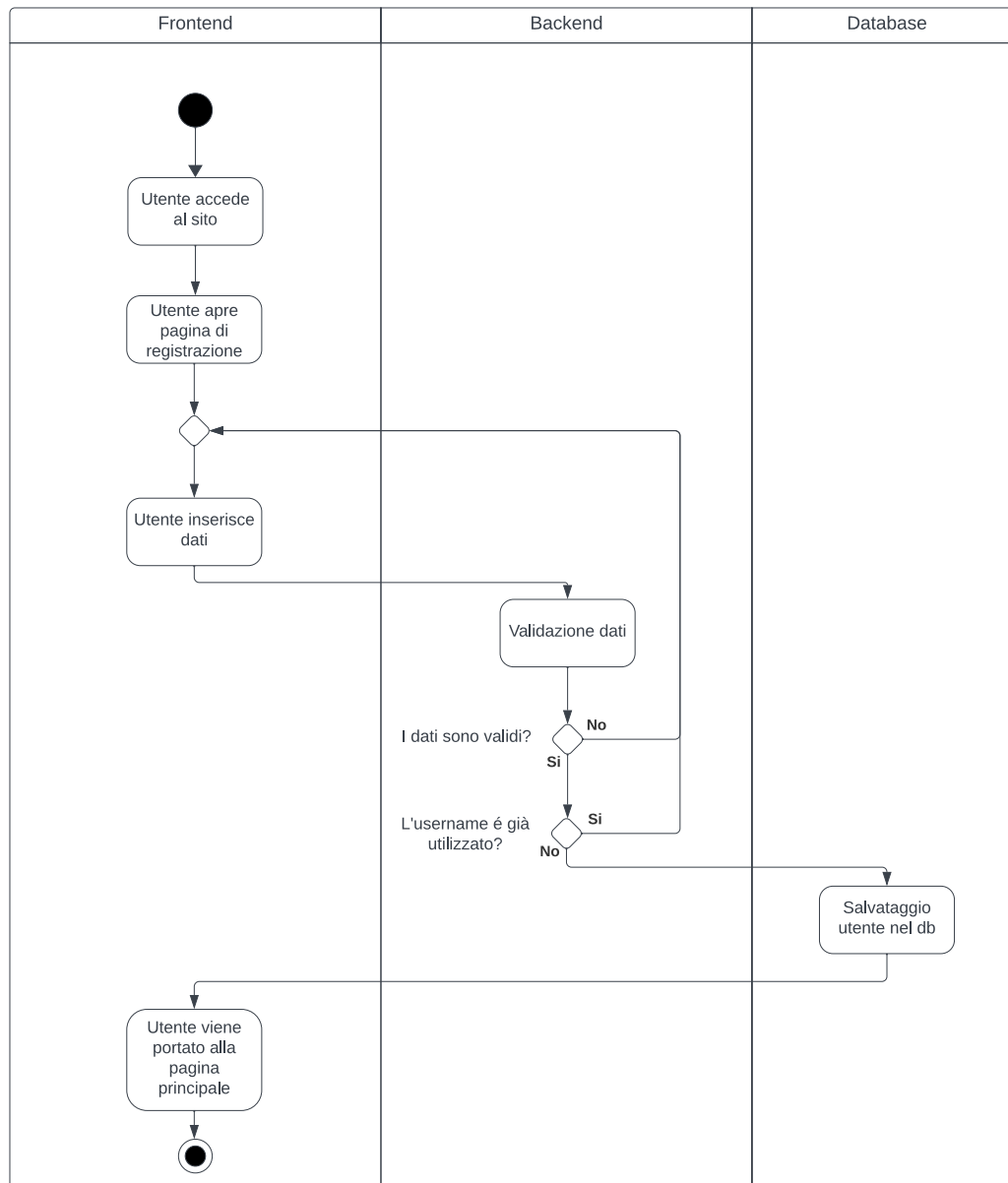


Figura 13 UML registrazione

Questo schema mostra il funzionamento della registrazione dell'utente sul sito. Il risultato che si vuole ottenere è quello di creare un account per l'utente in modo che possa salvare le proprie elaborazioni. Per farlo, una volta che l'utente ha inserito i propri dati per registrarsi (username e password) vengono validati e poi viene controllato se lo username è già stato utilizzato da qualcuno. Se i valori sono validi e l'username non è ancora utilizzato l'account viene creato e l'utente viene portato sulla pagina principale. Se invece non passa una delle due condizioni viene richiesto all'utente di inserire dei nuovi dati.

5 Implementazione

5.1 Backend

Per il backend ho deciso di utilizzare python (3.9.18) con il framework Flask. Ho deciso di utilizzare python con questo framework perché lo conoscevo già. Il framework Flask mi ha permesso di creare una REST API per comunicare con il frontend. Oltre alla REST API, per poter comunicare con il frontend ho anche utilizzato Socket.IO in modo da poter comunicare senza dover aspettare una richiesta da parte del frontend.

Per interfacciarmi con il database ho deciso di utilizzare l'ORM SQLAlchemy anche in questo caso perché lo conoscevo. Il database che andrò ad utilizzare sarà un database MySQL.

5.1.1 Struttura Cartelle

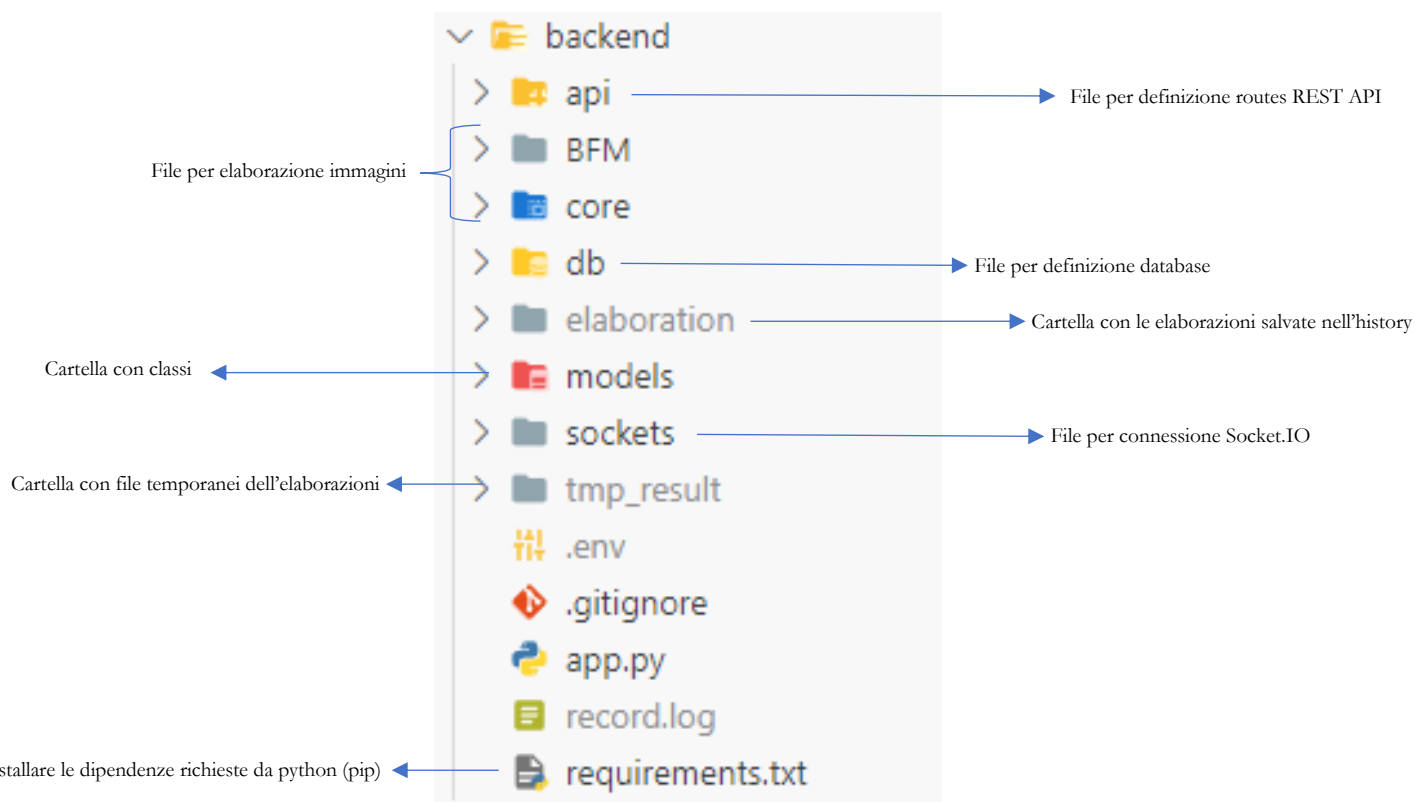


Figura 14 Struttura cartelle Backend

5.1.2 Database

Per potersi collegare al database per prima cosa bisogna configurare l'applicazione Flask passandogli la stringa di connessione per il database. La stringa di connessione è salvata all'interno di un file env.

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

Il database di questo sito è composto da due tabelle (come visto dal diagramma ER): la tabella User e la tabella Elaboration.

5.1.2.1 User

```
class User(db.Model):
    username = db.Column(db.String(20), primary_key=True)
    password = db.Column(db.String(64).with_variant(CHAR(64), "mysql", "mariadb"))
    token = db.Column(db.String(72).with_variant(CHAR(72), "mysql", "mariadb"))
    token_creation = db.Column(db.DateTime)
    elaborations = db.relationship('Elaboration', backref='user')

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

Questa è la dichiarazione della tabella User. Questa tabella ha le proprietà username, password, token e token_creation. Il campo elaborations serve come riferimento per poi creare la foreign key tra le due tabelle. Oltre alle proprietà è anche presente la funzione __repr__ per stampare l'oggetto. Questa classe possiede anche delle funzioni per gestire l'autenticazione sul sito che spiegherò in seguito.

5.1.2.2 Elaboration

```
class Elaboration (db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    user_fk = db.Column(db.String(20), db.ForeignKey('user.username'))
    folder_path = db.Column(db.String(20))
    creation_date = db.Column(db.DateTime)

    def __repr__(self):
        return f'<Elaboration id: {self.id} folder: {self.folder_path} date: {self.creation_date}>'
```

Questa è la dichiarazione della tabella Elaboration. Questa tabella ha come proprietà un ID, un riferimento all'utente che ha creato l'elaborazione (la foreign key), la path dove vengono salvate le elaborazioni e la data di creazione. Anche questa classe possiede la funzione __repr__ per poter stampare l'oggetto. Inoltre, questa classe possiede anche una funzione che serve per convertire l'oggetto elaboration in un JSON contenente anche l'immagine e il modello 3D per poter inviare l'elemento via REST API.

```
def convert_to_json(self):
    full_path = os.path.join(PATH, self.folder_path)
    full_path_img = os.path.join(full_path, "img.jpg")
    full_path_mesh = os.path.join(full_path, "model.glb")
    img = Image.open(full_path_img)
    img_bytes = io.BytesIO()
    img.save(img_bytes, format=img.format)
    img = encodebytes(img_bytes.getvalue()).decode('ascii')
    with open(full_path_mesh, "rb") as f:
        encodedZip = b64encode(f.read())
        mesh = encodedZip.decode()
    return {
        "id": self.id,
        "user": self.user_fk,
        "img": img,
        "mesh": mesh,
        "date": self.creation_date,
        "mesh_name": self.folder_path
    }
```

5.1.3 REST API

La REST API viene utilizzata per far comunicare il backend con il frontend. Tutte le richieste ritornano un JSON e uno status code. In certi casi i JSON che vengono inviati come risposta sono vuoti perché basta avere lo status code al frontend per eseguire delle operazioni senza dei valori aggiuntivi.

5.1.3.1 Autenticazione

Per gestire l'autenticazione vengono utilizzati dei token che vengono poi salvati nel database assieme alla loro data di creazione. Il token viene inviato all'utente quando fa login sul sito e viene salvato all'interno dei cookies. Questo token viene poi rinviato al backend quando si esegue una richiesta in modo da potersi autenticare.

```
def create_auth_token(self):
    salt = bcrypt.gensalt()
    token = bcrypt.hashpw(self.username.encode('utf-8'), salt)
    self.token = token
    self.token_creation = datetime.now()
    db.session.commit()
```

Questa funzione serve per creare il token per autenticarsi. I token vengono creati utilizzando un metodo di salting per renderli più sicuri. Quando il token viene creato viene salvata nel database la data di creazione.

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

Questa funzione viene richiamata prima di eseguire ogni richiesta che necessita l'autenticazione. Se l'autenticazione è valida, viene salvato all'interno di una variabile globale l'oggetto user per poter accedere alle informazioni dell'utente che ha fatto la richiesta dalle altre funzioni.

```
@staticmethod
def verify_auth_token(token):
    user = User.query.filter_by(token=token).first()
    if not user:
        logger.info(f'token {token} is not valid')
        return False
    if user.token_creation < datetime.today() - timedelta(days=30):
        logger.info(f'token for user {user.username} expired, need to login')
        return False
    return user
```

Questa è la funzione che viene richiamata per verificare che il token sia valido. Per prima cosa viene controllato se esiste un utente che ha assegnato il token che bisogna verificare. Se l'utente esiste, prima di confermare che l'autenticazione è valida, viene a controllato che non siano passati più di 30 giorni dalla creazione del token. Se anche questa condizione viene soddisfatta viene ritornato l'oggetto dell'utente che ha eseguito l'autenticazione.

5.1.3.2 Routes

5.1.3.2.1 Login

La route raggiungibile su **/api/login** utilizza il metodo GET e serve per poter eseguire l'accesso sul sito. Questa route richiede l'autenticazione e, se questa è valida, ritorna il token all'utente in modo che per le richieste successive possa inviare al server il token e non l'username e la password.

5.1.3.2.2 Register

La route raggiungibile su **/api/register** utilizza il metodo POST e serve per poter creare un account sul sito. Per poter creare un account si necessita di un nome utente e una password. Il nome utente deve essere lungo almeno cinque caratteri e al massimo lungo venti e deve anche essere univoco. La password invece deve avere almeno sei caratteri, contenere almeno un numero e almeno un carattere speciale. Per verificare tutti questi criteri viene utilizzata una regular expression.

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

Se tutti i criteri, sia per il nome utente che per la password, vengono soddisfatti viene creato un nuovo utente all'interno del database.

5.1.3.2.3 User

La route raggiungibile su **/api/user** utilizza il metodo GET e serve semplicemente per ottenere le informazioni dell'utente avendo il token. Questa route viene richiamata appena si apre la pagina web e si ha un token salvato all'interno dei cookies.

5.1.3.2.4 History

La route raggiungibile su **/api/history** utilizza il metodo GET e serve per accedere ai propri valori salvati all'interno dell'history. Questa route viene chiamata appena si apre la pagina, nel caso si avesse effettuato l'accesso. Questa route semplicemente controlla che l'autenticazione sia valida e nel caso ritorna i valori appartenenti all'utente salvati nell'history.

Per raccogliere i dati dall'history viene preso l'username dell'utente che ha fatto la richiesta e viene utilizzato per fare una query al database per prendere tutte le elaborazioni fatte dall'utente. Le elaborazioni vengono ordinate dalla più recente alla meno recente.

```
def get_history_elements(self, username):  
    with self.app.app_context():  
        elements=db_models.Elaboration.query  
        .filter_by(user_fk= username)  
        .order_by(desc(db_models.Elaboration.creation_date))  
        history_arr = []  
        for el in elements:  
            history_arr.append(el.convert_to_json())  
        return history_arr
```

5.1.4 Socket.IO

Per tutta la parte di comunicazione legata all'elaborazione dei modelli 3D ho deciso di utilizzare Socket.IO invece che usare la REST API perché in questo modo posso comunicare al frontend lo stato dell'elaborazione senza dover aspettare una richiesta da parte di questo.

```
@socketio.on('process_image')
def handle_user_image(data):
    logger.info(f"new elaboration received for id {request.sid}", request.remote_addr)
    try:
        image_data = re.sub('^data:image/.+;base64,', '', data["img"])
        img = Image.open(BytesIO(base64.b64decode(image_data)), mode='r')
        img = ImageOps.exif_transpose(img)
        img = img.convert('RGB')
        emit('image', {'status': EVENT_OK})
        logger.info(f"image from id {request.sid} is valid", request.remote_addr)
    except:
        emit('image', {'status': EVENT_NOT_OK})
        logger.warning(f"image from id {request.sid} is not valid", request.remote_addr)
    return
    qm.add_to_queue(request.sid, img, data["auth"])
```

Il backend ascolta un solo evento di Socket.IO ovvero l'evento che avvia il processo di elaborazione. Questo evento si occupa di verificare che si riesca ad accedere all'immagine passata dall'utente. Se si riesce viene aggiunta alla coda di elaborazione l'immagine assieme all'id della richiesta di Socket.IO in modo che si possa rispondere all'utente corretto con l'immagine corretta. Inoltre, se viene fornito, nell'elemento in coda viene salvato anche il token d'autenticazione dell'utente per salvare l'elaborazione nel database.

Per controllare sia valida viene convertito l'immagine da base64 a un oggetto Image. Se non viene lanciata alcuna eccezione vuol dire che l'immagine è valida. Prima di iniziare l'elaborazione vengono fatte alcune modifiche alla foto:

- Viene controllato se nei metadati exif è salvato una rotazione dell'immagine. Se è presente l'immagine viene ruotata come salvato nei metadati. Quest'azione viene eseguita perché se la faccia è girata di 90° non viene riconosciuta.
- L'immagine viene convertita in RGB. Questo viene fatto per prevenire lo sfondo trasparente delle immagini PNG andando così a rimuovere il valore dell'Alpha e trasformando così lo sfondo trasparente in bianco.

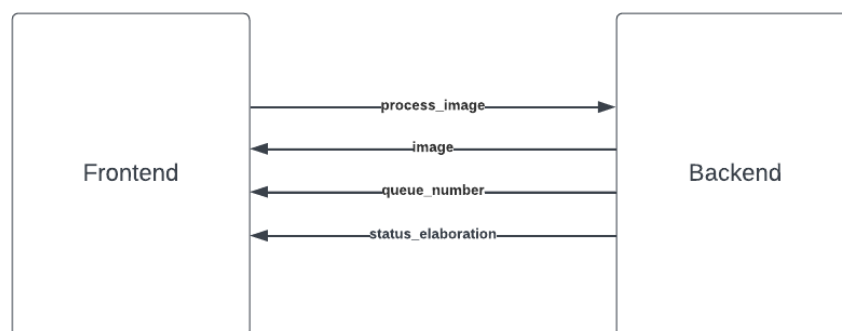


Figura 15 Comunicazione Socket.IO

5.1.4.1 Gestione coda

Essendo che il processo di creazione di un modello 3D partendo da un'immagine è abbastanza pesante, ho pensato di creare una coda in modo che viene eseguita un'elaborazione alla volta e non più di una contemporaneamente.

```
class QueueValue:
    def __init__(self, id, img, auth):
        self.id = id
        self.img = img
        self.auth = auth
        self.done = False

    def __repr__(self) -> str:
        return f"<Value with id {self.id} >"
```

All'interno della coda vengono salvati istanze di questa classe, ovvero oggetti che hanno come proprietà l'id della connessione socket, l'immagine da elaborare, il token d'autenticazione dell'utente e una variabile booleana per sapere se l'elemento è già stato elaborato.

Per gestire questa coda ho creato una Thread separata da quella di Flask.

```
def start_process(self):
    queue_thread = Thread(target=self.process_queue)
    queue_thread.daemon = True
    queue_thread.start()
```

Per poter eseguire una Thread con Flask bisogna avviarla come daemon altrimenti non funziona come dovrebbe. Questa Thread esegue una funzione che ogni secondo controlla se ci sono elementi all'interno della coda e se al momento c'è già un elemento che sta venendo elaborato. Se nessun elemento sta già venendo elaborato viene preso l'elemento più vecchio all'interno della coda e viene elaborato all'interno di un'altra Thread.

Questa funzione controlla anche che all'interno della coda non ci siano elementi che sono già stati elaborati, e se sono presenti verranno rimossi.

5.1.5 Elaborazione modelli

L'elaborazione dei modelli è la parte del progetto che mi ha preso più tempo. Ho fatto diversi tentativi ottenendo scarsi risultati, e quando sono riuscito a trovare un metodo funzionante, questo necessitava di un sistema operativo Linux e di una scheda grafica per poter funzionare e inizialmente non l'avevo disponibile. Infine sono riuscito a trovare una macchina che soddisfaceva i requisiti richiesti per poter far andare il metodo di elaborazione. Per l'elaborazione ho utilizzato le librerie pytorch e pytorch3D che, installando il CUDA Toolkit si poteva utilizzare la scheda video per poter processare l'immagine per poter calcolare la profondità e creare un modello 3D. Per l'installazione delle dipendenze è presente nel README.md una guida per installarle.

5.1.5.1 Inizializzazione modelli

Quando viene avviata l'applicazione, viene controllato se la macchina che la esegue ha installato CUDA. Dopodiché vengono creati due modelli di Machine Learning:

- Un modello pre-allenato per riconoscere e ritagliare una faccia da una foto
- Un modello che serve a fare delle predizioni sulla profondità dell'immagine

```
self.logger.info(f"Loading models for elaboration")
self.mtcnn = MTCNN(device=self.device, select_largest=False)
self.fa = face_alignment.FaceAlignment(face_alignment.LandmarksType, flip_input=False)

model_path = 'BFM/BFM09_model_info.mat'
model_dict = loadmat(model_path)
self.recon_model=BFM09ReconModel(model_dict,device=self.device,batch_size=1,
img_size=self.tar_size)
self.logger.info(f"Models loaded succesfully")
```

5.1.5.2 Ricerca della faccia

Come detto in precedenza, per andare a trovare viene utilizzato un modello pre-allenato. Se non viene trovata alcuna faccia l'elaborazione si ferma e viene comunicato al frontend che non è stata trovata alcuna faccia. Se invece la faccia viene trovata, questa viene ritagliata.

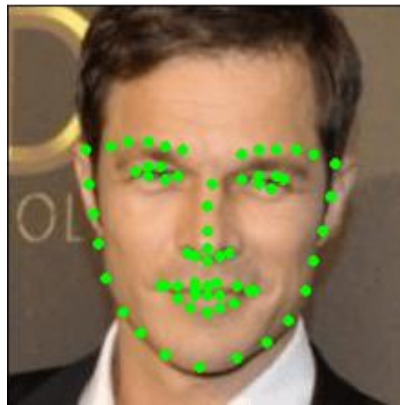


Figura 16 Riconoscimento volto per l'elaborazione

Una volta ritagliata la faccia vengono salvati in una lista i landmarks (i punti di riferimento) ovvero i punti verdi nell'immagine sopra, che vengono utilizzati poi per creare il modello 3D.

5.1.5.3 Creazione modello

Adesso che si sono ottenuti i punti di riferimento per poter ricreare la faccia vengono eseguiti dei calcoli per poter capire come è ruotata la faccia e come è posizionata. Una volta ottenuti questi valori si passa alla parte effettiva dove viene creato il modello 3D facendo delle predizioni. Una volta finite le predizioni vengono raccolti tutti i dati e viene creata e salvata la maschera, ovvero il modello 3D.

5.1.5.4 Conversione modello 3D

L'elaborazione genera un file .obj però per mostrarlo sul sito c'è stato bisogno di convertirlo nel formato glb.

```
path = os.path.join(self.res_folder, name+'_mesh.obj')
n_path = os.path.join(self.res_folder, name+'_mesh.glb')
if self.validate_obj_file(path):
    os.remove(path)
    return "INVALID-VALUES"
scene = a3d.Scene.from_file(path)
scene.save(n_path)
os.remove(path)
return name+'_mesh.glb'
```

5.1.5.5 Salvataggio dati

Quando si fa eseguire un'elaborazione e si è autenticati, l'immagine iniziale e il risultato finale vengono salvati per poterli visualizzare all'interno dell'history. I due file (immagine e modello 3D) vengono salvati all'interno del server nella cartella *elaboration* e poi viene salvato nel database un record per sapere dove sono salvati. Per identificare le diverse elaborazioni i due file vengono salvati all'interno di una sottocartella chiamata con uno uuid.

```
folder = uuid.uuid4().hex
parent = os.getcwd()
parent = os.path.join(parent, 'elaboration')
path = os.path.join(parent, folder)
os.mkdir(path)
```

Questo è il codice che crea la cartella chiamata con lo uuid. Questo uuid viene poi salvato all'interno nel database come folder_path.

5.1.6 Gestione errori

Essendo che sono stati implementati due metodi di comunicazione gli errori vengono comunicati in due metodi diversi.

Per gli errori della REST API gli errori vengono comunicati con un messaggio ed il codice d'errore 400 o 401 (usato solo per il login fallito).

```
return {'msg': 'password-invalid'}, 400
```

Invece per gli errori di Socket.IO, ogni messaggio che viene inviato il server, contiene la proprietà *status* che può contenere i valori “OK” oppure “NOK”. Se il valore di status è “NOK” vuol dire che il messaggio è un errore. Oltre alla proprietà *status* ci può anche essere un messaggio che comunica qual è il problema.

```
self.socketio.emit('status_elaboration', {"status": self.msg_not_ok, "msg": mesh_name})
```

5.2 Frontend

Per il frontend ho deciso di utilizzare il framework Sveltekit (2.0) utilizzando Typescript. Ho scelto di utilizzare questo framework principalmente perché volevo imparare ad utilizzare un nuovo framework. Insieme a Sveltekit ho utilizzato la libreria Skeleton che utilizza Tailwind e fornisce diversi componenti per la grafica.

5.2.1 Struttura Cartelle

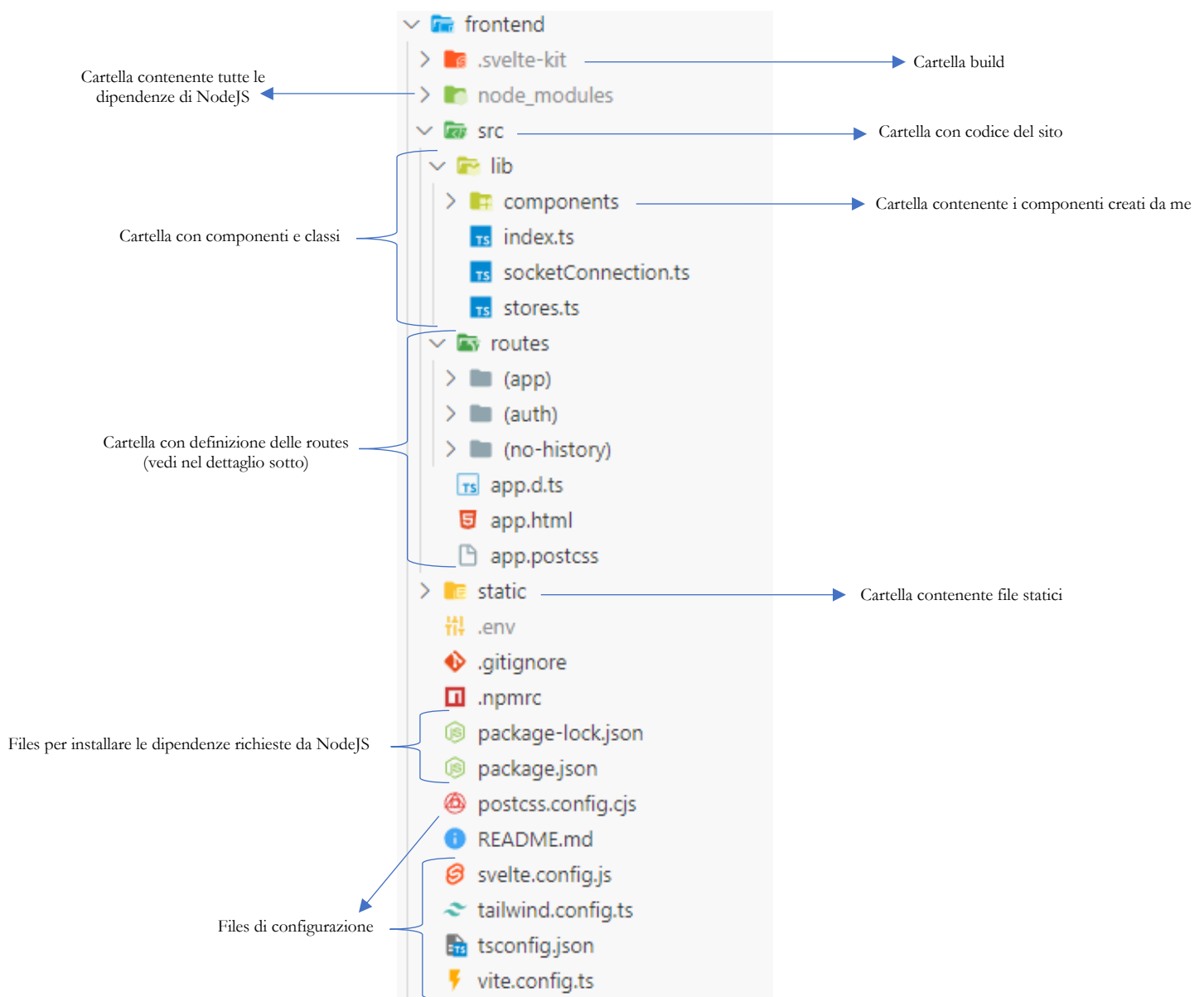


Figura 17 Struttura cartelle frontend

5.2.2 Pagine e navigazione

Per lo stile del sito web ho utilizzato un tema messo a disposizione da Skeleton. Il tema che ho selezionato utilizza la seguente palette di colori.



Figura 18 Palette dei colori

5.2.2.1 Navigazione

All'interno del sito web sono presenti diverse pagine: la pagina principale, la pagina di login, la pagina di registrazione, la pagina d'attesa dell'elaborazione, la pagina di visualizzazione del modello 3D e la pagina dell'history (per la visualizzazione dell'history da mobile).

Per la navigazione tra le diverse pagine Sveltekit utilizza un *filesystem-based router* ovvero le route vengono definite dalle cartelle all'interno del progetto.

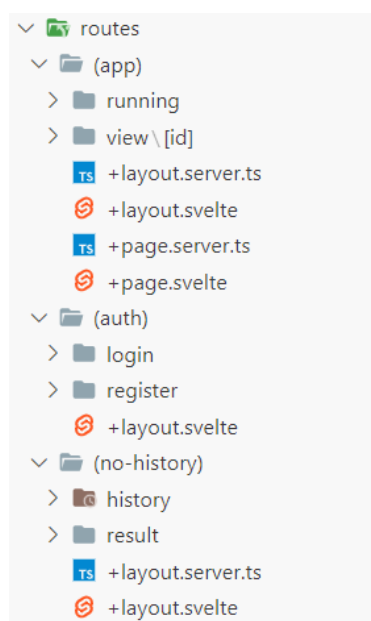


Figura 19 Definizione Routes

Come si vede dal contenuto della cartella *routes* al suo interno sono presenti i 3 gruppi *app*, *auth* e *no-history*. Ogni gruppo contiene al suo interno un file che definisce il layout che devono avere le pagine al suo interno.

Quindi la pagina principale (che si trova direttamente all'interno del gruppo *app*), la pagina d'attesa e la pagina di visualizzazione di un valore nell'history hanno un layout, le pagine di login e di registrazione un altro e la pagina dell'history e quella del risultato di elaborazione un altro ancora. In più ogni pagina o layout può avere un file **.server.ts* che può servire per fetchare dei dati oppure per gestire le azioni dei form.

5.2.2.2 Pagine di autenticazione

Le pagine di autenticazione (Login e registrazione) sono molto simili. L'unica cosa che cambia è che nella pagina di registrazione è presente un campo di testo in più per la conferma della password.

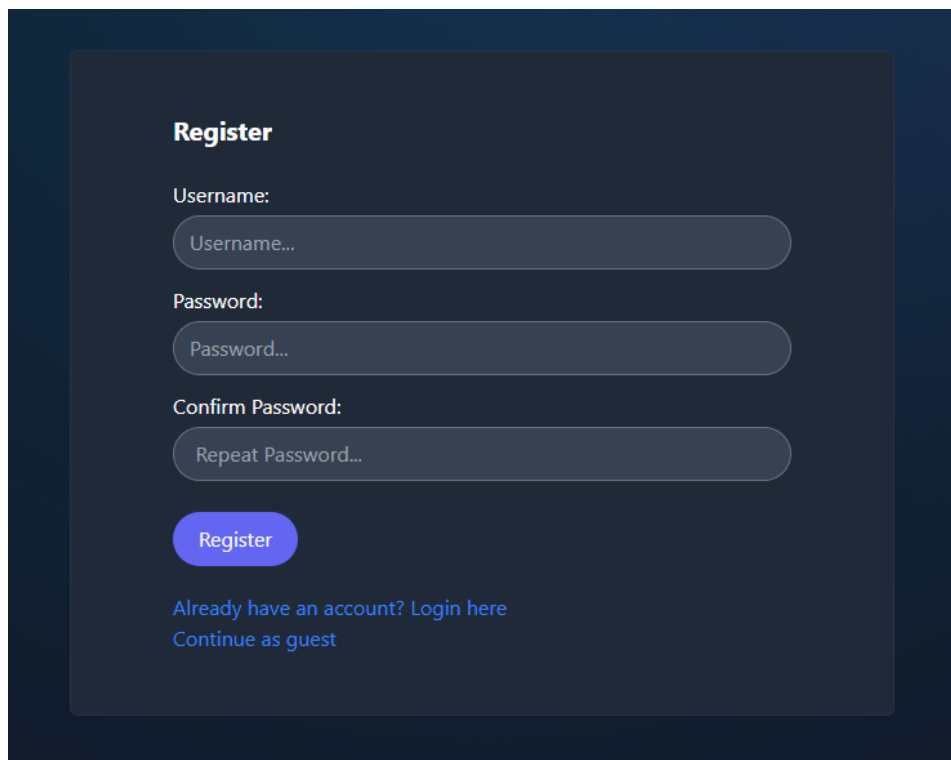


Figura 20 Pagina di registrazione

L'username deve avere almeno 5 caratteri, al massimo 20 e non deve essere già utilizzato. La password invece deve essere lunga almeno 6 caratteri, deve contenere almeno un numero e un carattere speciale.

Se una di queste condizioni non viene rispettata verrà mostrato all'utente un messaggio d'errore.

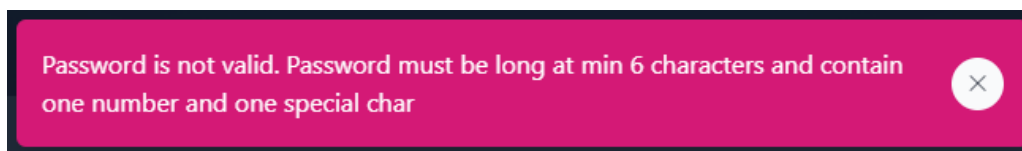


Figura 21 Messaggio d'errore

5.2.2.3 Pagina principale

Nella pagina principale è presente un header da dove si può poi accedere alle due pagine di autenticazione.



Figura 22 Header pagina principale

Nella parte centrale sono presenti due campi: uno per poter fare l'upload di un'immagine da andare ad elaborare e un secondo che serve per aprire la webcam. Alla destra invece è presente l'history.

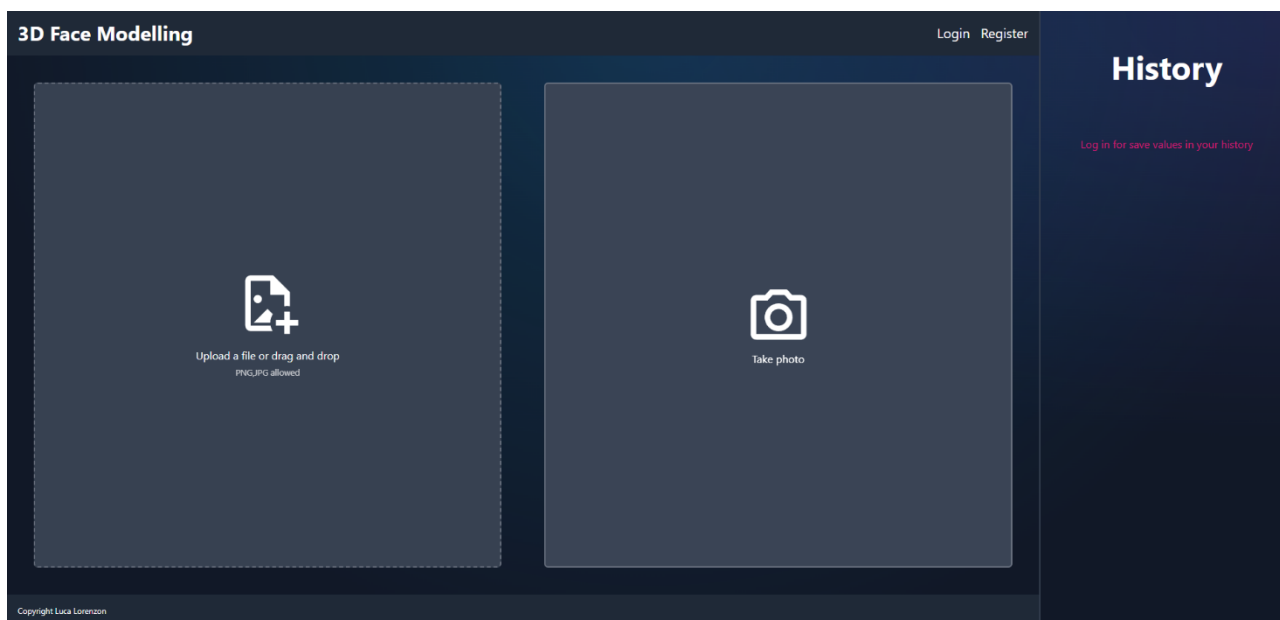


Figura 23 Pagina principale

Quando si apre la webcam viene prima di tutto chiesto il permesso di accedere alla webcam. Se il permesso non viene dato verrà mostrato un messaggio d'errore.

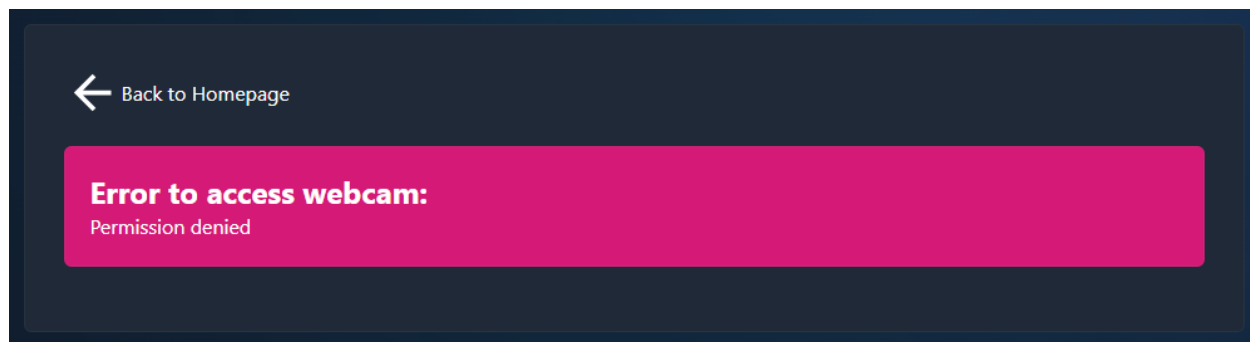


Figura 24 Errore permessi non dati

Se invece si dà il permesso la webcam si aprirà. Sopra alla webcam sarà presente un pulsante per chiudere la webcam e sotto invece è presente un pulsante per scattare la foto.

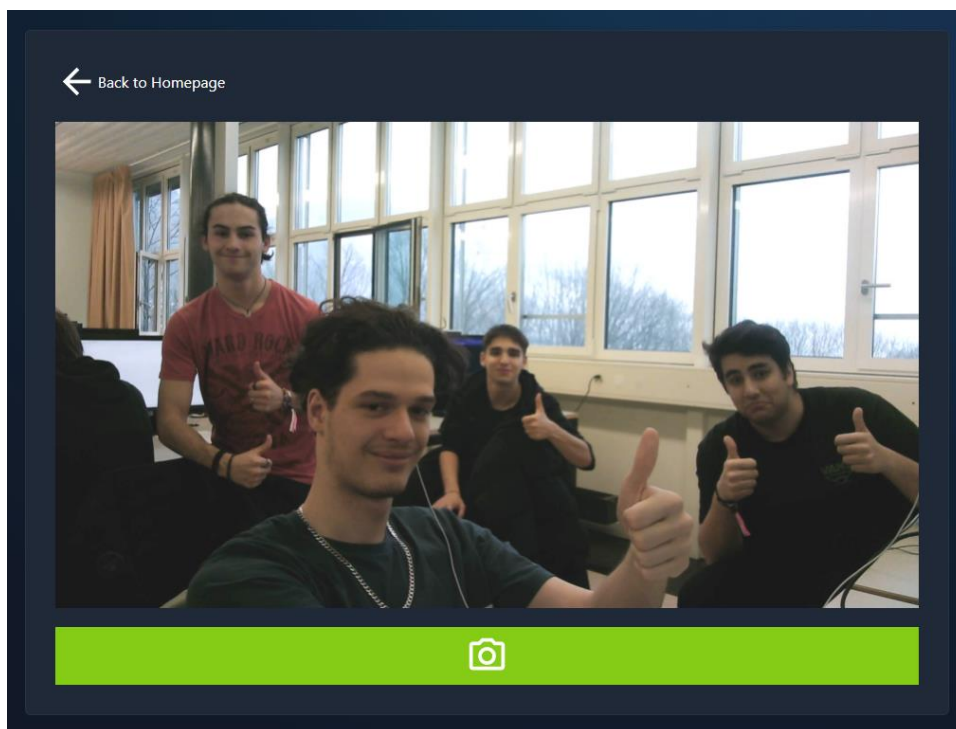


Figura 25 Webcam

Una volta selezionata una foto (sia scattata sul momento con la webcam che caricata) verrà mostrata nella seguente schermata dove si può cambiare foto oppure inviarla al server.

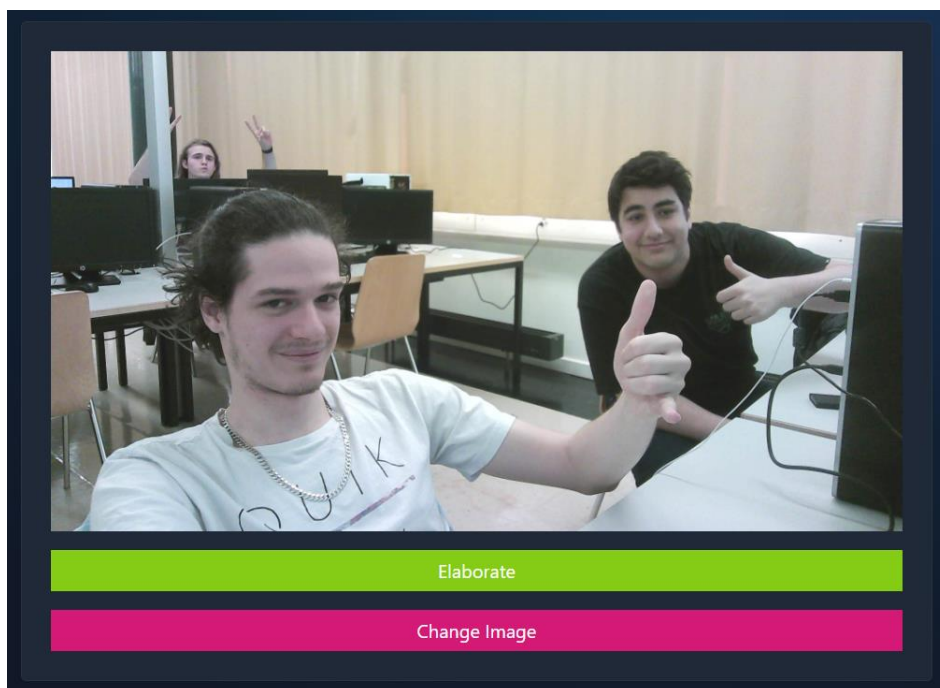


Figura 26 Foto selezionata

5.2.2.3.1 Cattura immagine

Per poter catturare delle immagini tramite la webcam ho creato un componente che gestisce la webcam.

```
async function obtenerVideoCamera(): Promise<void> {
    try {
        loading = true;
        stream = await navigator.mediaDevices.getUserMedia({
            video: {
                width: { ideal: 1920 },
                height: { ideal: 1080 }
            },
        });
        videoSource.srcObject = stream;
        videoSource.play();
        loading = false;
    } catch (error: any) {
        webcamErrorMsg = error.message;
        webcamError = true;
        loading = false;
    }
};
```

Grazie a questa funzione si può aprire la webcam. Nel caso non venga dato il permesso viene lanciato un errore e il messaggio d'errore viene mostrato all'utente.

```
function capturePhoto(): void {
    const canvas = document.createElement("canvas");
    canvas.width = videoSource.videoWidth;
    canvas.height = videoSource.videoHeight;
    canvas.getContext("2d")!.drawImage(videoSource, 0, 0, canvas.width, canvas.height);
    img = canvas.toDataURL("image/jpeg");
    closeStream();
}
```

Per scattare un'immagine viene chiamata questa funzione che salva il frame al momento che viene cliccato il pulsante all'interno del canvas e il contenuto del canvas viene salvato nell'URL dell'immagine. Una volta salvata l'immagine la webcam viene chiusa.

5.2.2.4 Pagina d'attesa

La pagina d'attesa è molto semplice: infatti è presente solamente una barra di caricamento dove poi verrà mostrato il progresso dell'elaborazione.

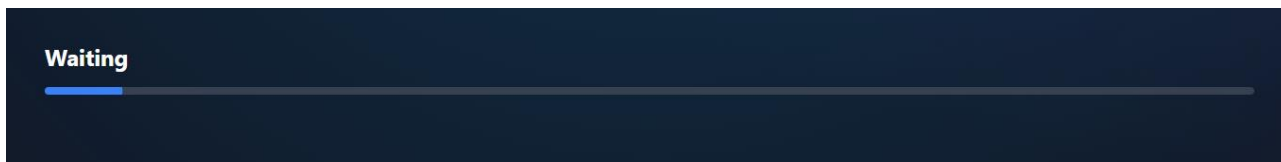


Figura 27 Pagina d'attesa

5.2.2.5 Pagina di visualizzazione

Nella pagina di visualizzazione si può vedere il risultato dell'elaborazione. Il modello è visualizzabile in tutte le direzioni ruotandolo. Inoltre si può anche scaricare il modello 3D nel formato glb.

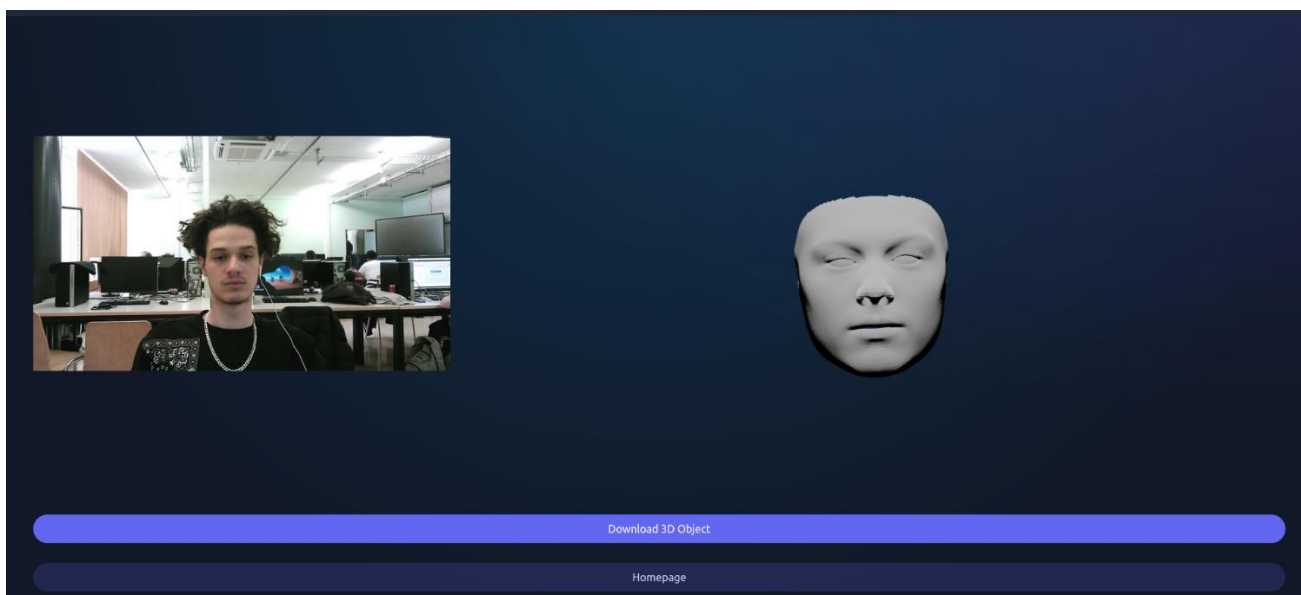


Figura 28 Pagina Visualizzazione

5.2.2.6 History

Come detto in precedenza l'history si trova sulla destra della pagina principale. Gli elementi vengono salvati nell'history soltanto se si ha eseguito l'accesso e un'elaborazione verrà mostrata nell'history con l'immagine originale e la data e l'ora di quando si ha fatto l'elaborazione.

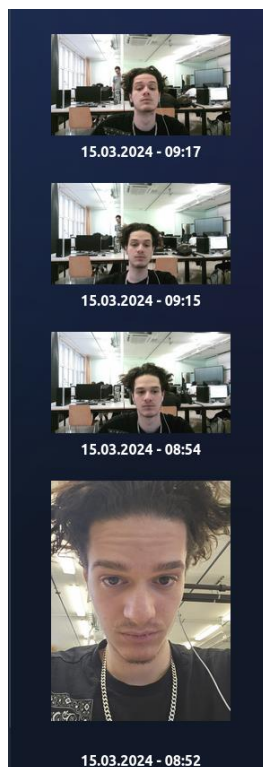


Figura 29 History

Quando si clicca su un elemento dell'history questo verrà selezionato e si aprirà al posto della pagina principale. Anche da questa schermata sarà possibile scaricare il modello 3D.

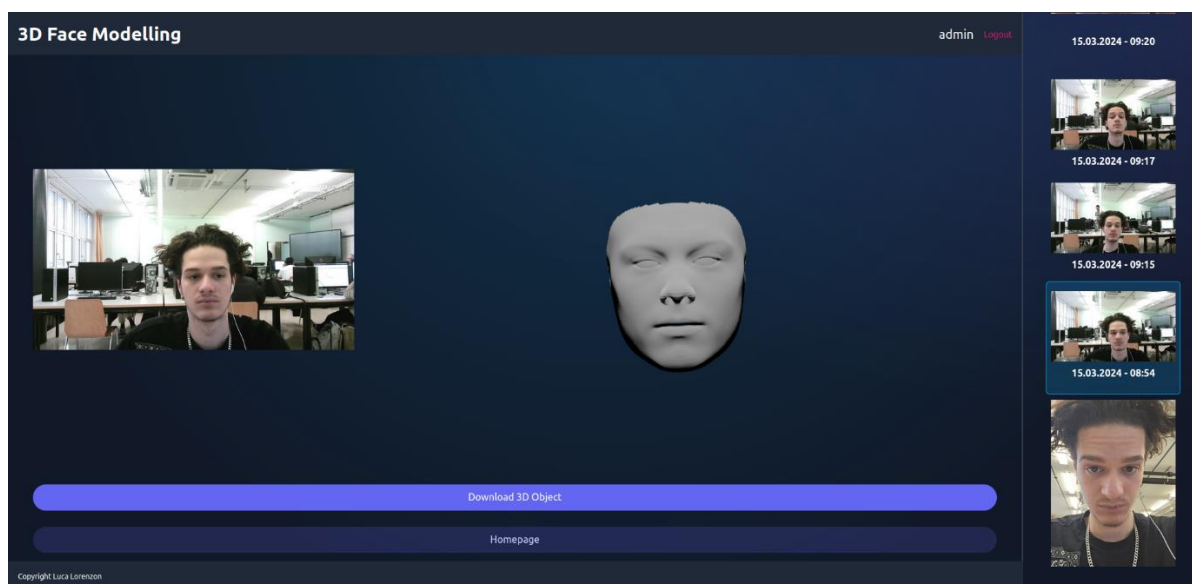


Figura 30 Elemento history

5.2.2.7 Responsive

Nel caso in cui si apra il sito web da un telefono oppure con una finestra troppo piccola la schermata principale subirà delle modifiche. Infatti l'history non sarà più presente nella schermata principale ma sarà possibile accederci tramite un pulsante sul fondo della pagina. Inoltre anche il pulsante per la webcam non sarà visibile perché quando si seleziona un file viene data come opzione di aprire la webcam quindi non serve un tasto apposta sul sito.

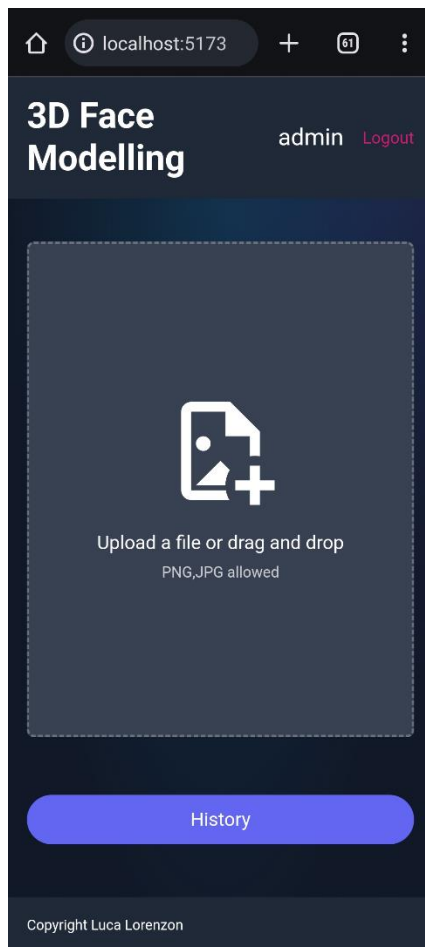


Figura 31 Pagina principale su telefono

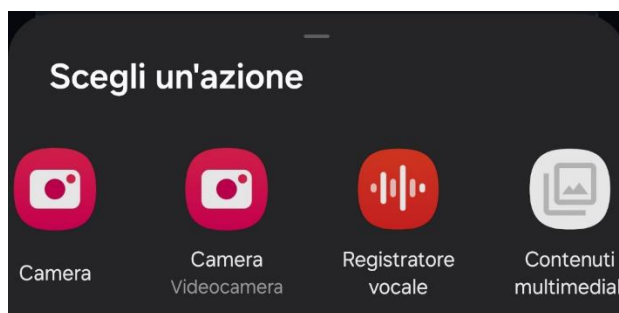


Figura 32 Menu upload immagine

5.2.3 Comunicazione con backend

5.2.3.1 REST API

Per poter comunicare con il backend principalmente viene utilizzata una REST API. Per poter fare le chiamate ho utilizzato la libreria “axios” che permette di fare delle richieste http.

```
const url = PUBLIC_BACKEND_URL + '/api/register';
const config: AxiosRequestConfig = {
  method: 'post',
  url: url,
  headers: { "Content-Type": "multipart/form-data" },
  data: {
    username: username,
    password: password
  },
  httpsAgent: new https.Agent({
    rejectUnauthorized: false
  })
}
const res: AuthResponse = await axios(config)
  .then((res) => {
```

Ho deciso di utilizzare questa libreria rispetto a delle opzioni già integrate in Javascript (come, per esempio, la funzione “fetch”) principalmente perché converte in automatico le risposte in JSON e perché ha integrato una sua gestione degli errori.

Se devo fare una richiesta che utilizza il token come autenticazione semplicemente viene impostato come username il token dell’utente.

```
let configHistory: AxiosRequestConfig = {
  method: 'get',
  url: urlHistory,
  auth: {
    username: token,
    password: 'unused'
  },
  httpsAgent: new https.Agent({
    rejectUnauthorized: false
  })
}
```

5.2.3.2 Socket.IO

Per poter aggiornare la barra di caricamento vengono utilizzati 2 eventi di Socket.IO. L'evento `queue_number` serve per aggiornare il numero di utenti che sono in coda.

```
io.on('queue_number', (res) => {
  console.log(res);
  const i = Number(res.index)
  queueIndex = i;
  if(i == 0 && !queueFinished){
    queueFinished = true;
  }
})
```

Invece l'evento `status_elaboration` serve per aggiornare la percentuale di elaborazione e per cambiare pagina a fine elaborazione. In quest'evento viene anche gestito se avviene un errore durante l'elaborazione.

```
io.on('status_elaboration', (res: any) => {
  if(res.status == responseOk){
    const status = Number(res.value);
    progressValue = status;
    if(status == 100){
      resultElaborationImg.update((v) => "data:image/jpeg;base64," + res.img)
      resultElaborationMesh.update((v) => "data:content/type;base64," + res.mesh)
      meshName.update((v) => res.mesh_name)
      goto("/result");
    }
  }else{
    let msg = 'Error while elaborating image.'
    if(res.msg == "NO-FACE"){
      msg += "Cannot find a face inside the image."
    }else if (res.msg == "INVALID-VALUES"){
      msg += "Cannot elaborate this image, retry again."
    }
    const t: ToastSettings = {
      message: msg,
      timeout: 5000,
      background: 'variant-filled-error',
    }
    toastStore.trigger(t);
    goto('/');
  }
})
```

5.2.4 Certificati

Per poter accedere alla webcam sul sito ho dovuto utilizzare dei certificati ssl self-signed. Per dire al sito di dover utilizzare i certificati ho dovuto modificare il file di configurazione di vite.

```
export default defineConfig({
  server: {
    https: {
      key: '../3dfacemodelling-privateKey.key',
      cert: '../3dfacemodelling.crt',
    }
  },
  plugins: [sveltekit(), purgeCss()]
});
```

In più ho dovuto specificare per ogni comunicazione con il backend di accettare le risposte anche da server con certificati self-signed essendo che di default vengono scartate.

```
httpsAgent: new https.Agent({
  rejectUnauthorized: false
})
```

```
export const io = ioClient(PUBLIC_BACKEND_URL, { secure: false });
```

6 Test

6.1 Protocollo di test

Test Case Riferimento	TC-001 REQ-001	Nome	Registrazione account
Descrizione	Bisogna registrare un nuovo account sul sito.		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Andare nella pagina di registrazione 		
Procedura	<ol style="list-style-type: none"> 1. Inserire un nome utente e due volte la password 2. Premere il tasto per registrarsi 		
Risultati attesi	L'account dovrà venir creato e si verrà portati nella pagina principale. In alto a destra, al posto dei pulsanti di registrazione e login, si potrà vedere il proprio username.		

Test Case Riferimento	TC-002 REQ-001	Nome	Registrazione account non valida
Descrizione	Bisogna registrarsi nel sito con dei valori non validi.		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Andare nella pagina di registrazione 		
Procedura	<ol style="list-style-type: none"> 1. Inserire nome utente e password non validi 2. Premere il tasto per registrarsi 		
Risultati attesi	Dovrà comparire un avviso che ricorda all'utente quali sono le convenzioni di username e password.		

Test Case Riferimento	TC-003 REQ-002	Nome	Login
Descrizione	Bisogna accedere al sito web con un account esistente		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Andare nella pagina di login 		
Procedura	<ol style="list-style-type: none"> 1. Inserire nome utente e password di un account esistente 2. Premere il tasto per accedere 		
Risultati attesi	Verrà aperta la pagina principale e si potrà accedere all'history dell'account. In più in alto a destra sarà mostrato il nome utente di chi ha fatto l'accesso.		

Test Case Riferimento	TC-004 REQ-002	Nome	Login non valido
Descrizione	Bisogna accedere al sito web con delle credenziali non valide.		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Andare nella pagina di login 		
Procedura	<ol style="list-style-type: none"> 1. Inserire delle credenziali di un account non valide 2. Premere il tasto per registrarsi 		
Risultati attesi	Dovrà comparire un avviso che dice all'utente che il login è fallito.		

Test Case Riferimento	TC-005 REQ-004	Nome	Upload immagine
Descrizione	Bisogna caricare un'immagine sul sito		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 		
Procedura	<ol style="list-style-type: none"> 1. Trascinare un'immagine nel Drag & Drop 		
Risultati attesi	L'immagine verrà mostrata a schermo e se si invia l'immagine al server verrà elaborata.		

Test Case Riferimento	TC-006 REQ-004	Nome	Cattura immagine via webcam
Descrizione	Bisogna catturare un'immagine con la webcam per poi poterla elaborare		
Prerequisiti	1. Aprire il sito web		
Procedura	1. Aprire la webcam dal sito 2. Scattarsi una foto		
Risultati attesi	L'immagine appena catturata con la webcam verrà mostrata sul sito e se la si invia al server verrà elaborata.		

Test Case Riferimento	TC-007 REQ-005	Nome	Visualizzazione history
Descrizione	Bisogna visualizzare i dati salvati nell'history		
Prerequisiti	1. Aprire il sito web		
Procedura	1. Eseguire l'accesso con delle credenziali valide 2. Cliccare su un valore dell'history.		
Risultati attesi	Si dovranno visualizzare tutte le elaborazioni fatte nella destra della pagina principale. Se si cliccherà su un'elaborazione verrà aperta e si potrà vedere meglio. Se l'account non ha mai elaborato delle immagini apparirà un messaggio al posto dell'history.		

Test Case Riferimento	TC-008 REQ-006	Nome	Elaborazione immagine
Descrizione	Bisogna elaborare un'immagine per creare un modello 3D		
Prerequisiti	1. Aprire il sito web		
Procedura	1. Caricare una foto sul sito o scattarla con la webcam 2. Inviarla al server per farla elaborare		
Risultati attesi	Sul sito dovrà essere mostrato il progresso dell'elaborazione.		

Test Case Riferimento	TC-009 REQ-007	Nome	Visualizzazione modello 3D
Descrizione	Bisogna visualizzare il risultato dell'elaborazione direttamente dal sito.		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Caricare un'immagine 		
Procedura	<ol style="list-style-type: none"> 1. Inviare l'immagine al server per farla elaborare 2. Visualizzare il risultato 3. Ruotare il modello 3D in tutte le direzioni 		
Risultati attesi	Il modello 3D si potrà ruotare in tutte le direzioni direttamente dal sito		

Test Case Riferimento	TC-010 REQ-007	Nome	Download modello 3D
Descrizione	Bisogna scaricare un modello 3D creato		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Caricare un'immagine e farla elaborare 		
Procedura	<ol style="list-style-type: none"> 1. Premere il tasto di Download sotto il risultato dell'elaborazione 		
Risultati attesi	Si dovrà scaricare un file preferibilmente di tipo OBJ o STL		

Test Case Riferimento	TC-011 REQ-008	Nome	Applicazione texture
Descrizione	Bisogna scaricare un modello 3D creato		
Prerequisiti	<ol style="list-style-type: none"> 1. Aprire il sito web 2. Caricare un'immagine e farla elaborare 		
Procedura	<ol style="list-style-type: none"> 1. Oltre a creare un modello 3D il server ci applicherà ad esso una texture estrapolata dall'immagine. 		
Risultati attesi	Il modello 3D elaborato avrà anche una texture basata sull'immagine principale.		

Test Case Riferimento	TC-012 REQ-009	Nome	Protocollo di log
Descrizione	Bisogna verificare che il protocollo di log funzioni facendo diverse operazioni.		
Prerequisiti	1. Aprire il sito web		
Procedura	1. Registrare un nuovo profilo 2. Eseguire l'accesso con un account 3. Elaborare delle immagini 4. Visualizzare dei dati nell'history.		
Risultati attesi	I log dovranno contenere delle informazioni riguardanti le azioni appena svolte.		

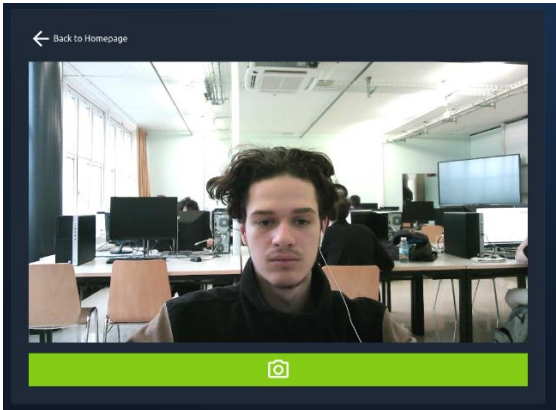


Test Case Riferimento	TC-013 REQ-010	Nome	Gestione coda elaborazioni
Descrizione	Bisogna verificare che la coda che gestisce le elaborazioni funzioni correttamente.		
Prerequisiti	1. Aprire il sito web su due finestre diverse		
Procedura	1. Caricare una foto sul sito sulla prima finestra 2. Far partire l'elaborazione dell'immagine 3. Caricare una foto sul sito sulla seconda finestra 4. Far partire l'elaborazione dell'immagine		
Risultati attesi	Sulla seconda finestra l'elaborazione non partirà subito, ma solo quando la prima è finita. Nel mentre verrà mostrato a schermo quanti utenti ci sono in coda.		

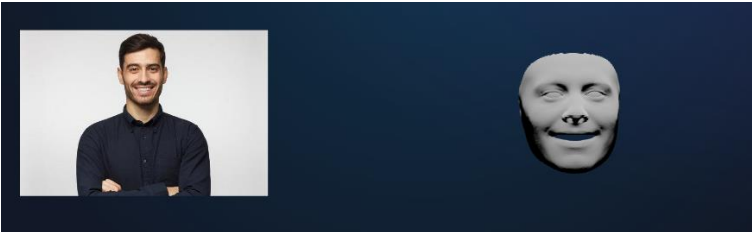

Test Case Riferimento	TC-014 REQ-006	Nome	Elaborazione immagini casi extra
Descrizione	Bisogna elaborare immagini con diverse condizioni per verificare il limite raggiungibile dal codice per creare un modello 3D.		
Prerequisiti	1. Aprire il sito web		
Procedura	<ol style="list-style-type: none"> 1. Elaborare foto con poca luce 2. Elaborare foto con tanta luce 3. Elaborare foto con occhiali 4. Elaborare foto con occhiali da sole 5. Elaborare foto con volto mezzo coperto 6. Elaborare foto con persone con età diverse 7. Elaborare foto con persone di etnie diverse 8. Elaborare foto con barba 9. Elaborare foto con sfondo trasparente 10. Elaborare foto con più persone 11. Elaborare foto con faccia capovolta 12. Elaborare foto contenente un disegno 		
Risultati attesi	Tutte le elaborazioni dovranno andare a buon fine		









6.2 Risultati test

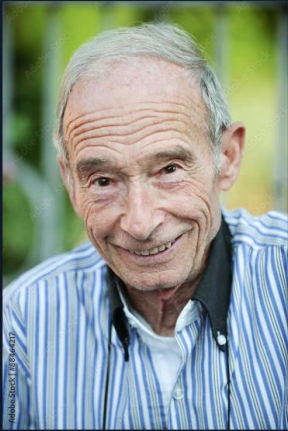
Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l'errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l'errore con eventuali ipotesi di correzione.





Test Case	Risultato ottenuto	Stato	Data
TC-001	Quando si registra un account con dei valori validi si viene portati alla pagina principale e in alto a destra apparirà il nome utente dell'account con cui ci si è registrati. 	Passato	20.03.2024
TC-002	Quando ci si registra con dei valori non validi i campi di testo vengono svuotati e in basso apparirà un messaggio d'errore. 	Passato	20.03.2024
TC-003	Quando si esegue Login all'interno del sito con delle credenziali valide, come quando si registra un account, si viene portati alla pagina principale e in alto a destra apparirà il nome utente.	Passato	20.03.2024
TC-004	Quando si accede al sito con delle credenziali non valide, i campi di testo del form di Login vengono svuotati e in basso apparirà un messaggio d'errore. 	Passato	20.03.2024
TC-005	Quando viene caricata un'immagine sul sito all'interno del drag & drop, questa verrà mostrata a schermo. 	Passato	20.03.2024

TC-006	<p>Quando si apre la webcam dal sito, verranno chiesti per prima cosa i permessi. Se vengono dati verrà avviata la cattura via webcam.</p> 	Passato	20.03.2024
TC-007	<p>Nella pagina principale si vedono le elaborazioni eseguite e quando si clicca su un'elaborazione questa si apre per poter vedere il modello 3D.</p> 	Passato	20.03.2024
TC-008	<p>Quando si invia un'immagine al server si vedrà a schermo il progresso dell'elaborazione tramite una barra di caricamento.</p> 	Passato	20.03.2024

TC-009	<p>Quando l'elaborazione finisce viene aperta una pagina dove si può visualizzare il risultato.</p> 	Passato	20.03.2024
TC-010	<p>Alla fine di un'elaborazione o quando si visualizza un'elaborazione è presente il tasto per poter scaricare il modello 3D. il modello 3D verrà scaricato in formato glb.</p>	Passato	20.03.2024
TC-011	<p>L'applicazione della texture al modello 3D non è stata implementata.</p>	Fallito	20.03.2024
TC-012	<p>All'interno del terminale verranno stampati diversi messaggi per mostrare le operazioni eseguite. I log vengono salvati anche all'interno di un file chiamato "record.log".</p> <pre>[INFO] - PERFORMANCE - getting history for user testcase: 7 ms [INFO] - 127.0.0.1 - [20/Mar/2024 09:04:13] new elaboration received for id qvWZrAyBqII4X1tAAAB [INFO] - 127.0.0.1 - [20/Mar/2024 09:04:13] image from id qvWZrAyBqII4X1tAAAB is valid [INFO] - SERVER - [20/Mar/2024 09:04:13] elaboration with id qvWZrAyBqII4X1tAAAB add to queue [INFO] - SERVER - [20/Mar/2024 09:04:13] No elaboration are running [INFO] - SERVER - [20/Mar/2024 09:04:13] starting to elaborate image with id qvWZrAyBqII4X1tAAAB [INFO] - SERVER - [20/Mar/2024 09:04:13] Starting to elaborate [INFO] - SERVER - [20/Mar/2024 09:04:14] Face is detected. l: 775, t: 330, r: 1193, b: 748 [INFO] - SERVER - [20/Mar/2024 09:04:17] start rigid fitting 100% 1000/1000 [00:07:00:00, 130.62it/s] [INFO] - SERVER - [20/Mar/2024 09:04:24] done rigid fitting. lm loss: 0.0004270678327884525 [INFO] - SERVER - [20/Mar/2024 09:04:24] Rotation [-0.01062127 0.0019214 -0.00504363] [INFO] - SERVER - [20/Mar/2024 09:04:24] Translation [-0.0108606 -0.06186641 1.2438908] [INFO] - SERVER - [20/Mar/2024 09:04:24] start non-rigid fitting 100% 500/500 [01:03:00:00, 7.87it/s] [INFO] - SERVER - [20/Mar/2024 09:05:29] composed image is saved at /home/sandbox/Documents/3d-face-modelli ng/5 Sito o applicativo/backend/tmp result as 8f07f32e40e54bcea025ifac091655c3 mesh.glb [INFO] - SERVER - [20/Mar/2024 09:05:29] created folder d2c0dea02cbb48bf8a555001e70d5d67 for user testcase [INFO] - SERVER - [20/Mar/2024 09:05:29] Image of testcase saved in /home/sandbox/Documents/3d-face-modelli ng/5 Sito o applicativo/backend/elaboration/d2c0dea02cbb48bf8a555001e70d5d67/1mg.jpg [INFO] - PERFORMANCE - elaboration with id qvWZrAyBqII4X1tAAAB: 75479 ms</pre>	Passato	20.03.2024
TC-013	<p>Se si fanno partire più elaborazioni allo stesso momento verrà mostrato quanti utenti ci sono in coda mentre si attende il proprio turno per l'elaborazione.</p> 	Passato	20.03.2024

TC-014			22.03.2024
1			Passato
2			Passato
3			Passato
4			Passato
5	Volto non trovato		Fallito

6			Passato
7			Passato
8			Passato
9			Passato
Lo sfondo trasparente viene trasformato in sfondo bianco prima dell'elaborazione			

10	 	Passato
11	Volto non trovato	Fallito
12	 	Passato

Se nella foto sono presenti più volti viene preso un volto a random.

6.2.1 Errore non conosciuto

Mentre stavo facendo debug ho notato un problema che blocca le elaborazioni per sempre. Praticamente a volte il codice ritorna dei valori non validi (NaN) e questo viene salvato all'interno del modello per le elaborazioni, quindi smettono di andare. Il problema è che non sono riuscito a trovare il motivo per il quale questo viene causato e quindi non ho potuto fare un test case per questo. Come soluzione ho optato di annullare l'elaborazione, inviare un messaggio d'errore all'utente e di ricaricare i modelli quando avviene questo errore.

6.3 Mancanze/limitazioni conosciute

Purtroppo non ho avuto il tempo per poter implementare l'applicazione di una texture ai modelli 3D generati estrapolandola dall'immagine originale.

7 Consuntivo

Qua sotto è presente il Gantt consuntivo. Negli allegati si può anche trovare una foto della kanban board consuntiva.

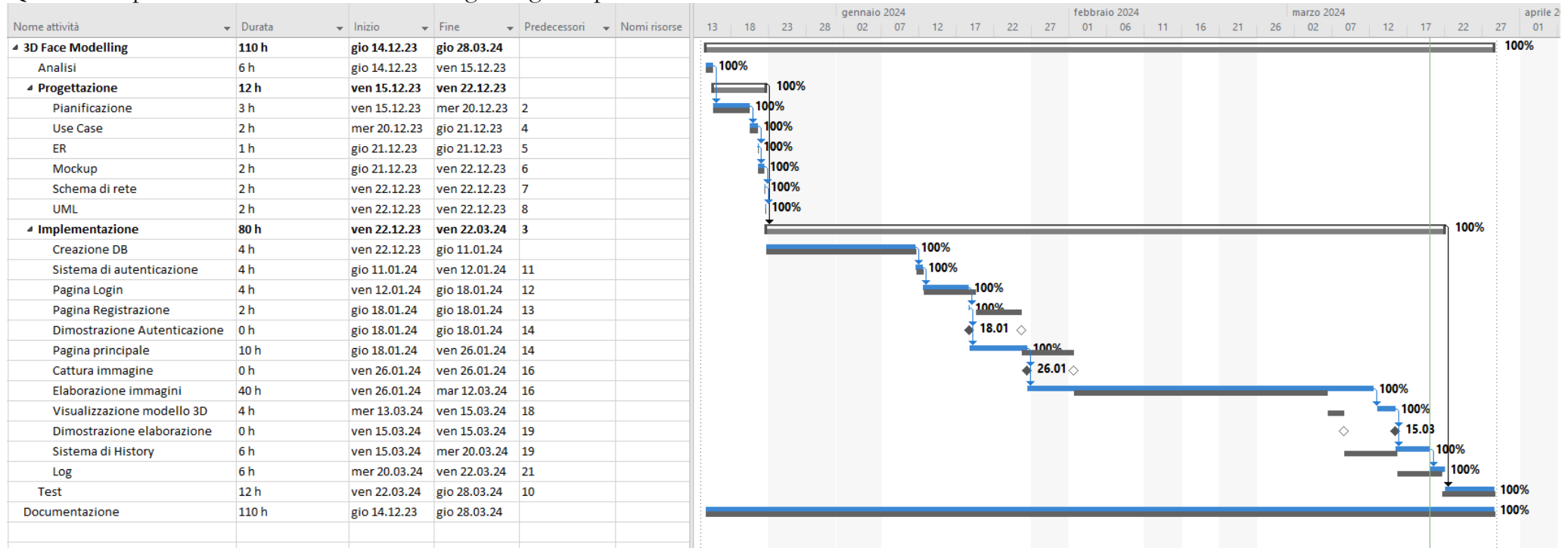


Figura 33 Diagramma di Gantt Consuntivo

7.1 Differenze con il Gantt preventivo

Come si vede dal diagramma di Gantt consuntivo ci sono state parecchie differenze rispetto a quello che avevo previsto:

- Sicuramente la differenza più grande è che è stata modificata la data di consegna del progetto. Questo è stato causato dallo spostamento della gita scolastica a dopo le vacanze di Pasqua, andando così ad anticipare la consegna di una settimana.
- Un'altra differenza è stata che ho perso parecchie ore a causa del reclutamento militare e per malattia.
- Andando ad analizzare invece proprio il lavoro svolto da me si può notare che avevo previsto più tempo di quello che necessitavo quasi per tutte le attività, tranne per la parte di elaborazione delle immagini che ho impiegato 10 ore in più del previsto. Però, appunto per il fatto che ho guadagnato tempo in quasi tutte le altre attività, sono riuscito comunque a stare bene dentro ai tempi messi a disposizione.
- Per poter starci completamente all'interno dei tempi ho dovuto dedicare un po' meno tempo al test e debug.

8 Conclusioni

Il progetto è stato completato con successo, infatti da una foto si riesce a creare un modello 3D. I modelli sono molti approssimati perché vengono creati basandosi solo sui lineamenti, quindi tutti i dettagli come barba, capelli, occhiali, etc. non vengono considerati. Comunque in generale è stato raggiunto l'obiettivo posto.

8.1 Sviluppi futuri

Per questo progetto ci potrebbero essere diverse migliorie come:

- Applicazione di una texture sui modelli 3D generati
- Eliminazione dei dati dall'history
- Poter selezionare quale faccia elaborare nel caso ce ne siano più di una nella foto
- Creare Pool di modelli già creati per sostituire dei modelli nel caso si blocchino
- Caricamento history in modo dinamico
- Migliorare la gestione della visualizzazione del progresso di elaborazione (renderlo più preciso)

8.2 Considerazioni personali

Arrivato alla fine del progetto posso ritenermi molto soddisfatto del risultato che sono riuscito ad ottenere. Anche se ho riscontrato parecchi problemi a implementare l'elaborazione delle immagini alla fine sono riuscito ad ottenere un buon risultato.

Grazie a questo progetto ho migliorato le mie conoscenze di python e del framework Flask, imparando anche qualcosa di nuovo come Socket.IO. Inoltre ho anche imparato ad utilizzare il framework Sveltekit e Typescript che non avevo mai utilizzato e con i quali mi sono trovato molto bene. Oltre a questo ho imparato anche ad utilizzare la libreria Three.js per poter mostrare i modelli 3D all'interno di un sito web. In generale mi sono trovato bene con tutte le tecnologie che ho utilizzato durante questo progetto.

9 Bibliografia

9.1 Sitografia

- <https://svelte.dev/>, *documentazione di svelte*, 22.12.2023
- <https://kit.svelte.dev/>, *documentazione di sveltekit*, 22.12.2023
- <https://www.skeleton.dev/>, *documentazione di skeleton*, 22.12.2023
- <https://tailwindcss.com/>, *documentazione di tailwind*, 22.12.2023
- <https://flask.palletsprojects.com/en/3.0.x/>, *documentazione di flask*, 22.12.2023
- <https://flask-sqlalchemy.palletsprojects.com/en/3.1.x/>, *documentazione di sqlalchemy*, 22.12.2023
- https://dev.to/nelsonmendezz_/how-to-use-the-webcam-with-svelte-js-2415, *guida webcam*, 19.01.2024
- <https://uiverse.io/>, *loaders open source fatti con CSS*, 19.01.2024
- <https://icon-sets.iconify.design/>, *icone*, 23.01.2024
- <https://python-socketio.readthedocs.io/en/stable/>, *documentazione socket.io*, 26.01.2024
- <https://pytorch.org/>, *documentazione pytorch*, 31.01.2024
- <https://pytorch3d.org/>, *documentazione pytorch3D*, 31.01.2024
- <https://datahacker.rs/007-3d-face-modeling-3dmm-model-fitting-in-python/>, *guida per elaborazione*, 09.02.2024
- <https://threlte.xyz/>, *documentazione threlte*, 13.03.2024
- <https://images.google.com/>, *immagini per test*, 22.03.2023

10 Glossario

Termine	Significato
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.
File env	Un file env è un file contenente delle configurazioni dell'applicazione, delle variabili d'ambiente o delle informazioni sensibili.
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.
Modello di Machine Learning	Un modello di Machine Learning è una struttura che impara delle informazioni da dei dati forniti. Con questi dati il modello cerca di trovarci una sequenza e prova a farci delle predizioni. Nel caso di un modello pre-allenato si intende un modello alla quale sono già stati dati una grande quantità di dati e quindi dovrebbe riuscire ad ottenere dei risultati validi
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.
Regular Expression	Una regular expression serve per vedere se una stringa soddisfa 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.
Salting	Il salting è una tecnica che aggiunge una stringa random ad un dato sensibile prima di andare ad applicarci un algoritmo di hash. In questo modo non si può fare reverse engineering sul dato cryptato per ottenere il dato sensibile.
SHA-256	Lo SHA-256 è un algoritmo di hash che ritorna una stringa di 64 caratteri.
Socket.IO	Socket.IO è una libreria che permette una comunicazione bidirezionale basata su degli eventi tra un client e un server.
Thread	Una thread è un processo che viene diviso dal processo principale per farlo andare in parallelo.
Token	Un token è una stringa di caratteri che viene utilizzata per autenticarsi senza dover utilizzare ad ogni richiesta le proprie credenziali.
WebSocket	WebSocket è una tecnologia che crea un canale di comunicazione bidirezionale su una singola connessione TCP.

11 Indice delle figure

Figura 1 Use Case	8
Figura 2 Kanban Board.....	9
Figura 3 Diagramma di Gantt	10
Figura 4 Schema di rete.....	14
Figura 5 Schema logico	15
Figura 6 Pagina di login.....	16
Figura 7 Pagina di registrazione	17
Figura 8 Pagina Principale	18
Figura 9 Pagina Principale con immagine caricata	18
Figura 10 Pagina Principale con elaborazione	19
Figura 11 Elemento dell'history	20
Figura 12 UML elaborazione	21
Figura 13 UML registrazione	23
Figura 14 Struttura cartelle Backend	24
Figura 15 Comunicazione Socket.IO	30
Figura 16 Riconoscimento volto per l'elaborazione.....	32
Figura 17 Struttura cartelle frontend.....	35
Figura 18 Palette dei colori.....	36
Figura 19 Definizione Routes	36
Figura 20 Pagina di registrazione	37
Figura 21 Messaggio d'errore	37
Figura 22 Header pagina principale.....	38
Figura 23 Pagina principale	38
Figura 24 Errore permessi non dati	38
Figura 25 Webcam.....	39
Figura 26 Foto selezionata.....	39
Figura 27 Pagina d'attesa.....	41
Figura 28 Pagina Visualizzazione.....	41
Figura 29 History	42
Figura 30 Elemento history	42
Figura 31 Pagina principale su telefono.....	43
Figura 32 Menu upload immagine.....	43
Figura 33 Diagramma di Gantt Consuntivo	59

12 Allegati

QdC	1_QdC\QdC 2Sem23-24_X_PeGe_3dFaceModelling.docx
Abstract	2_Abstract\Abstract.pdf
Diari di lavoro	4_Diari\
Schema database	7_Allegati\Database
Diagramma di Gantt	7_Allegati\Pianificazione
Kanban baord consuntiva	7_Allegati\Pianificazione
Diagrammi Swimlane	7_Allegati\Swimlane
Mockup	7_Allegati\Mockup
Schema di rete	7_Allegati\SchemaRete
Use Case	7_Allegati\UseCase
Worklog	7_Allegati\Worklog
Prodotto	5_Sito_o_applicativo
Setup per applicazioni	README.md