Esempio di documentazione

1 Introduzione 4

1.1 Informazioni sul progetto 4

1.2 Abstract 4

Situazione iniziale 4

Attuazione 4

Risultati 4

1.3 Scopo 4

2 Analisi 5

2.1 Analisi del dominio 5

2.2 Analisi e specifica dei requisiti 5

2.3 Use case 10

2.4 Pianificazione 10

2.5 Analisi dei mezzi 11

2.5.1 Software 11

2.5.2 Hardware 11

3 Progettazione 12

3.1 Design dell’architettura del sistema 12

3.2 Design dei dati e database 12

3.2.1 Diagramma E/R 12

3.1 Design delle interfacce 14

3.1.1 Pagina di login 14

3.1.2 Pagina per registrarsi 15

3.1.3 Pagina ricerca viaggi 16

3.1.4 Pagina mostra tratta 17

3.1.5 Pagina acquista biglietti 18

3.1.6 Pagina home amministratore 19

3.1.7 Pagina crea bus 20

3.1.8 Pagina crea tratta 21

3.1.9 Pagina gestione utenti 22

3.1.10 Pagina dettagli utente 23

3.1.11 Pagina fattura 24

3.1.12 Pagina fermate 25

3.1.13 Pagina passeggeri 26

3.2 Design procedurale 27

4 Implementazione 28

4.1 Struttura applicativo 28

4.2 Database 28

4.3 Pagina di home 29

4.4 Pagina mostra tratta 32

4.5 Pagina acquista biglietto 33

4.6 Pagina registrazione 34

4.7 Pagina di login 36

4.8 Pagina di home - amministratore 37

4.9 Pagina di creazione bus 38

4.10 Pagina di creazione viaggi 39

4.11 Pagina gestione utenti 40

4.12 Pagina informazioni utenti 41

4.13 Pagina fatture 42

5 Test 43

5.1 Protocollo di test 43

5.2 Risultati test 46

5.3 Mancanze/limitazioni conosciute 46

6 Consuntivo 46

7 Conclusioni 46

7.1 Sviluppi futuri 46

7.2 Considerazioni personali 46

7.2.1 Luca Fumasoli 47

8 Bibliografia 47

8.1 Bibliografia per articoli di riviste: 47

8.2 Bibliografia per libri 47

8.3 Sitografia 48

9 Glossario 48

# Introduzione

## Informazioni sul progetto

Allievi: Andrea Frati, Luca Fumasoli, Manuel Grosso

Classi: I3AC / I3AA

Docente responsabile: Guido Montalbetti

Data inizio: 03.02.2022

Data fine: 12.05.2022

## Abstract

**Situazione iniziale**

*Lo scopo di questo progetto è di creare un’applicazione web che possa aiutare le persone a trovare in modo semplice un viaggio che vogliono percorrere ad una data ed orario voluto. Una volta trovato il viaggio ricercato dovrà metterli a disposizione un metodo per pagare all’interno del sito con la scelta di un posto libero desiderato.*

*Quindi il risultato atteso è un sito per facilitare la ricerca e la riservazione dei viaggi alle persone.*

**Attuazione**

*Per questo progetto sarà necessario avere un account amministratore per creare viaggi ed i mezzi per i viaggi, e ci sarà bisogno di poter lasciar creare un account agli utenti del sito per poter permettergli di riservare posti sui viaggi desiderati. Per realizzare il sito abbiamo usato i principali linguaggi web, ossia HTML, CSS, PHP e Javascript.*

**Risultati**

*Breve analisi dei risultati ottenuti (obiettivi raggiunti, …), …*

## Scopo

Lo scopo del progetto è di creare un sito facile da navigare su cui riservare biglietti per viaggi che si è interessati a fare.

# Analisi

## Analisi del dominio

I viaggi all’interno del sito potranno essere visualizzati da chiunque e se sono interessati potranno creare un account per riservare un biglietto. Ci saranno anche degli account di amministratore che potranno creare viaggi ed i mezzi su cui percorrere i viaggi.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Creazione di un database |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Gestione di differenti livelli di utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | 3 livelli, amministratore, utente registrato e utente non registrato con annesso il salvataggio nel database  Ogni tipo di utente ha interfacce grafiche diverse. |
| **Sotto requisiti** | |
| **021** | Possibilità di registrarsi |
| **022** | Possibilità di fare login |
| **023** | Possibilità di distinguere i vari tipi di utente |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Interfacce grafiche differenti per i 3 tipi di utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Amministratore:   * Tutto quello che può fare un utente loggato * Gestire i partecipanti al tragitto * Stampare la lista dei partecipanti e filtrarla * Possibilità di stampare una fattura da inviare ai partecipanti o inviare un’email * Creare nuovi tragitti * Creare nuovi autobus * Inviare email agli utenti registrati con filtri   Utente registrato:   * Vedere i tragitti * Filtrare i tragitti e ordinarli * Vedere disponibilità di posti in un viaggio * Selezionare e prenotare uno o più posti in un viaggio   Utente non registrato:   * Vedere i tragitti * Filtrare i tragitti e ordinarli * Vedere disponibilità di posti in un viaggio * Registrarsi al sito * Fare login |
| **Sotto requisiti** | |
| **031** | Poter differenziare i 3 differenti utenti. |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Inserire nel database un tragitto |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Tragitto composto da:   * Località di partenza * Località di arrivo * Data partenza * Data arrivo * Fermate e partenze intermedie con gli orari * Costo del tragitto da partenza a arrivo * Costo per le differenti tappe |
| **Sotto requisiti** | |
| **041** | Possibilità di salvare e leggere le tratte su un database |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Controllare e gestire i partecipanti dei viaggi |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’amministratore può vedere tutti i partecipanti e cambiare i dati se non sono corretti. |
| **Sotto requisiti** | |
| **051** | Creazione dei viaggi |
| **052** | Utenti devono poter partecipare ai viaggi |
| **053** | Poter leggere dal database chi partecipa ai viaggi |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Stampare la lista di partecipanti |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’amministratore può stampare la lista dei partecipanti che può essere divisa per località  La lista indica per uno specifico luogo, dando orario e partenza, chi sale o scende e quale posto ha o aveva |
| **Sotto requisiti** | |
| **061** | Sistema di filtraggio per località |
| **062** | Stampare i dati dei partecipanti |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Sistema filtrabile di mass mail |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Mail da mandare in contemporanea a tutti i partecipanti selezionati. |
| **Sotto requisiti** | |
| **072** | Poter selezionare gli utenti |

|  |  |
| --- | --- |
| **ID: REQ-8** | |
| **Nome** | Gestione dei pagamenti |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’amministratore può vedere tutte le fatture degli utenti |
| **Sotto requisiti** | |
| **001** | Fattura del viaggio |
| **002** | Opzione di stampa fattura da inviare ai partecipanti |
| **003** | Opzione di invio email di fattura ai partecipanti |

|  |  |
| --- | --- |
| **ID: REQ-09** | |
| **Nome** | Ricerca tragitto |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Ci deve essere la possibilità di cercare i tragitti esistenti tramite inserimento di filtri: - orario di partenza - luogo di partenza e luogo di destinazione |
| **Sotto requisiti** | |
| **091** | Filtro sulla ricerca |

|  |  |
| --- | --- |
| **ID: REQ-10** | |
| **Nome** | Registrazione al sito |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Ci deve essere la possibilità di registrarsi al sito inserendo un’email e una password che poi verranno salvate nel database |
| **Sotto requisiti** | |
| **101** | Controllo della mail |
| **102** | Criptaggio delle password |

|  |  |
| --- | --- |
| **ID: REQ-11** | |
| **Nome** | Log-in al sito |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Ci deve essere la possibilità di fare il log-in al sito inserendo il proprio email e la propria password |
| **Sotto requisiti** | |
| **111** | Bisogna essere già registrati nel sito, quindi avere email e password salvati nel database |

|  |  |
| --- | --- |
| **ID: REQ-12** | |
| **Nome** | Prenotazione biglietto |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Se si è loggati al sito, dopo aver selezionato una tratta si può prenotare uno o più posti |
| **Sotto requisiti** | |
|  | Bisogna essere già registrati nel sito, quindi avere email e password salvati nel database |
|  | Bisogna avere la possibilità di selezionare una tratta |

**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.

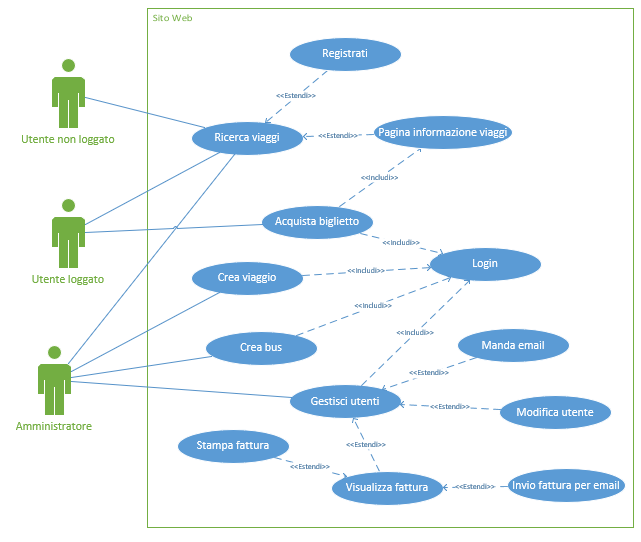
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

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.

## Use case



Ci sono 3 tipi di utente:

1. **Utente non loggato:** Può accedere alla pagina di ricerca viaggi ed usarla per trovare viaggia alla quale è interessato. Può inoltre visualizzare le informazioni delle tratte e se vuole può registrarsi per poter diventare un utente loggato.
2. **Utente loggato:** Può fare tutte le cose che può fare un utente non loggato ma se accede col proprio account all’applicativo dalla pagina informazione viaggi può acquistare un biglietto scegliendo il proprio posto per il viaggio sul bus.
3. **Amministratore:** Può fare tutto ciò che può fare l’utente loggato, ed inoltre può creare dei viaggi, creare dei bus e gestire gli utenti. Per le sue funzionalità di gestione utenti può visualizzare tutti gli utenti esistenti nel database, modificare i loro dati, visualizzare le fatture per i viaggi dove hanno acquistato un posto, e queste fatture possono stamparle od inviarle per email.

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

|  |
| --- |
| Figura : Esempio di diagramma di Gantt. |

Se si usano altri metodi di pianificazione (p.es. scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

### Software

I software che abbiamo usato sono:

* Microsoft office professional plus 2019:
  + Microsoft word: per fare la documentazione
  + Microsoft visio: per fare lo schema del sito e dei database
  + Microsoft Project: per fare il diagramma di Gantt
* Visual Studio Code versione 1.58.2
* XAMPP versione 3.3.0
* Google Chrome versione 96.0.4664.110
* PHP 7.2
* HTML 5
* CSS 3
* Javascript 1.8.5
* MySQL 8.0
* Librerie:
  + Bootstrap 4.3.1 (https://getbootstrap.com/)

### Hardware

Come hardware abbiamo usato solo i computer scolastici, le loro componenti sono:

* Processore: Intel(R) Xeon(R) CPU E3-1240 V2 @ 3.40GHz
* RAM: 8 GB
* Scheda video: NVIDIA Quadro 600

# Progettazione

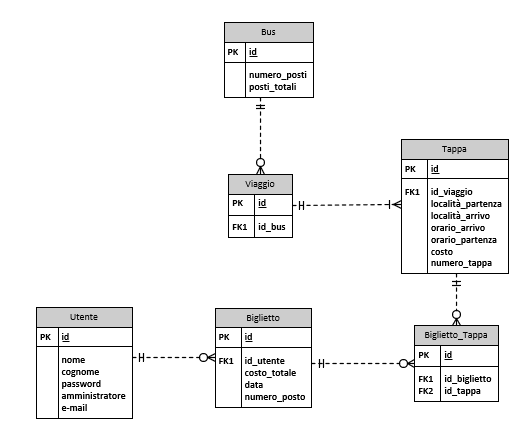
## Design dell’architettura del sistema

Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

### Diagramma E/R



Questo diagramma E/R mostra tutte le tabelle del database del nostro progetto con le primary e foreign key.

Tutti gli attributi in ogni tabella del database sono obbligatori alla creazione di un nuovo record ed ogni tabella ha un id come primary key.

#### Utente

I campi contenuti nell’entità utente sono il nome ed il cognome dell’utente, la password dell’account, l’email ed un booleano amministratore che con un valore di 1 rende l’account un amministratore e con il valore di 0 un utente normale.

#### Biglietto

L‘entità biglietto contiene una foreign key per sapere che utente ha acquistato il biglietto, un attributo per il costo totale del biglietto, uno per la data del viaggio ed un ultimo per sapere che posto è stato prenotato dall’utente.

#### Bus

L’entità bus contiene solo 2 campi oltre la primary key, il numero di posti totali del bus ed il numero della tratta che percorre il bus.

#### Viaggio

L‘entità viaggio ha solo la propria primary key ed una foreign key per sapere che bus percorre questo viaggio.

#### Tappa

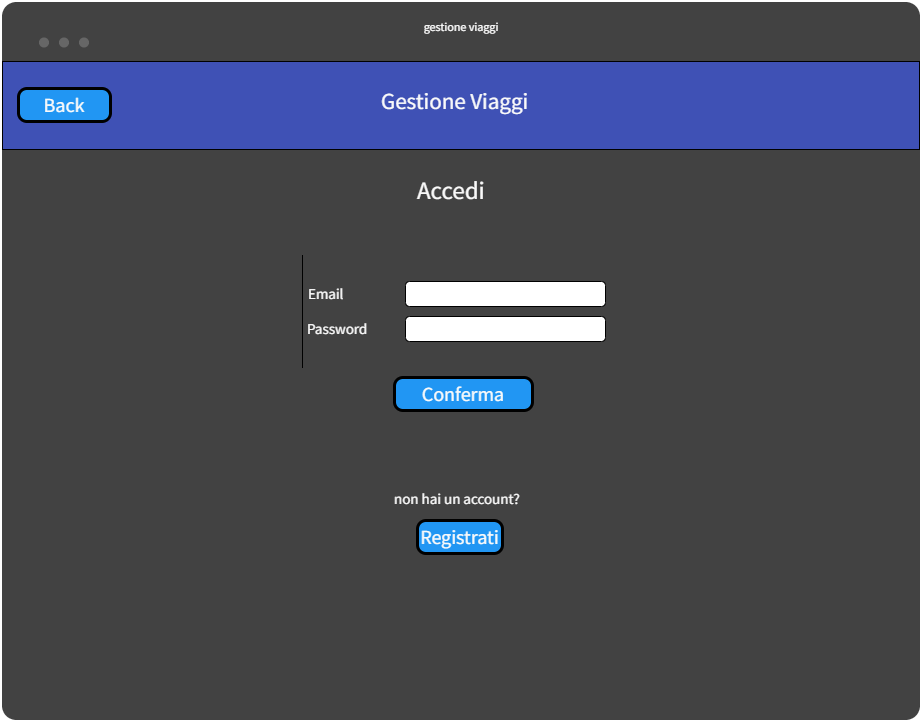
L‘entità tappa contiene tutte le fermate di un viaggio. Ha un attributo per la località di partenta ed un altro per quella di arrivo, e ne ha altri due per sapere a che ora parte dalla località di partenza ed a che ora arriva alla destinazione. Ha anche un campo per il costo della tappa, uno per il numero della tappa ed una foreign key per sapere di che viaggio è la tappa,

#### Biglietto\_Tappa

La tabella biglietto\_tappa è una relazione tra biglietto e tappa e contiene solo due foreign key, ossia l’id del biglietto riservato per la tappa ed un id per sapere quale è la tappa. Questa tabella sarà popolata da vari record per ogni biglietto per sapere su quali tappe vale il biglietto.

## Design delle interfacce

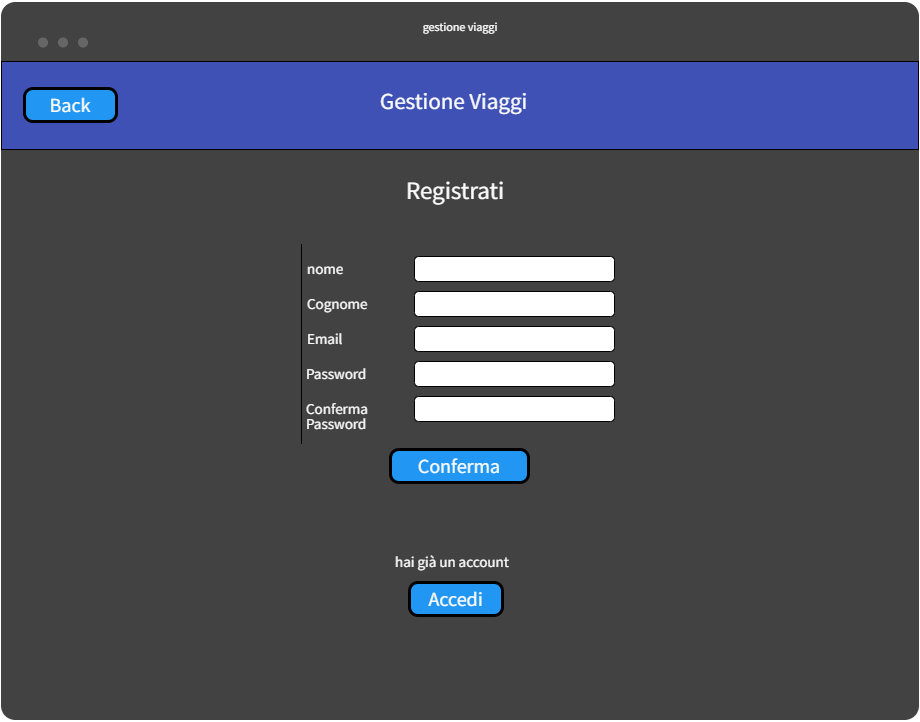
### Pagina di login



Pagina per accedere al proprio account immettendo la propria e-mail e password. Se non si ha creato un account si può premere su registrati per creare un account.

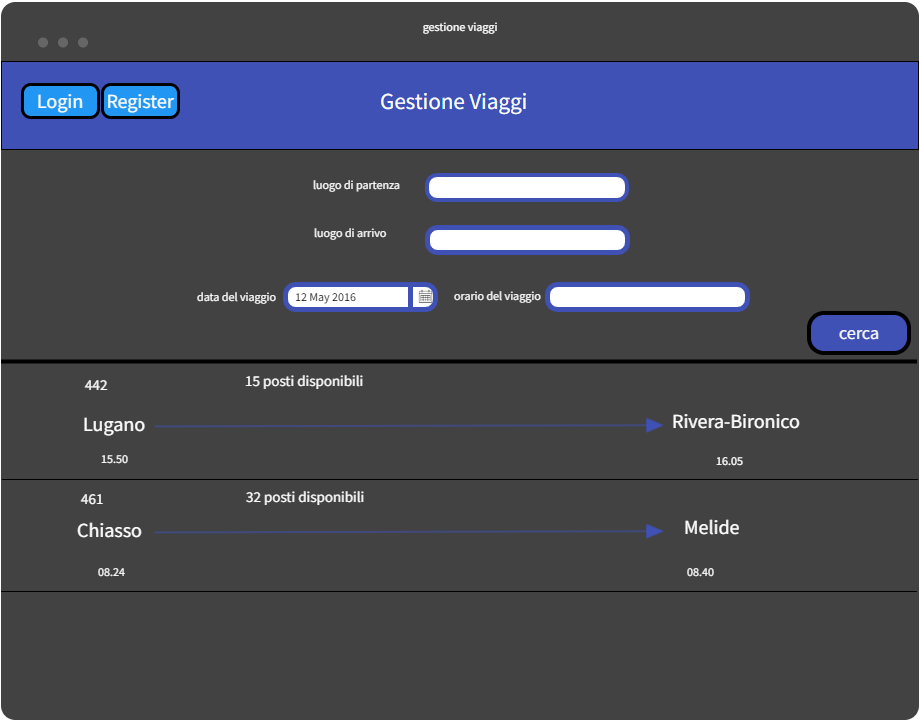
login.

### Pagina per registrarsi



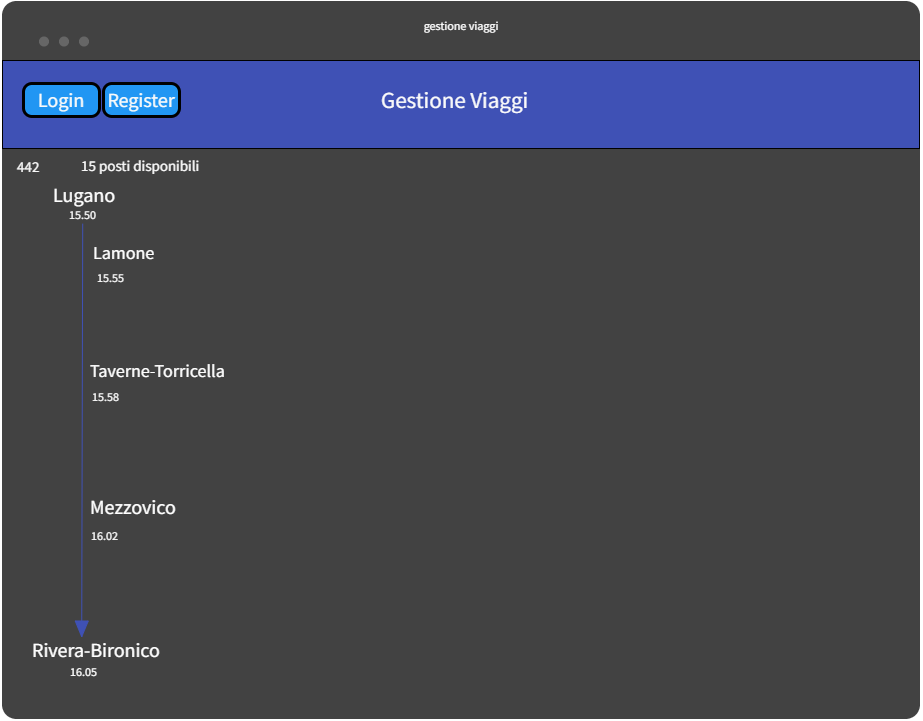
Pagina per creare il proprio account immettendo nome, cognome, E-mail e 2 volte la password per essere sicuri che non si fanno errori e poi premendo su conferma viene creato l’account. Se si ha già un account si può premere su accedi e ti porta alla pagina di login.

### Pagina ricerca viaggi



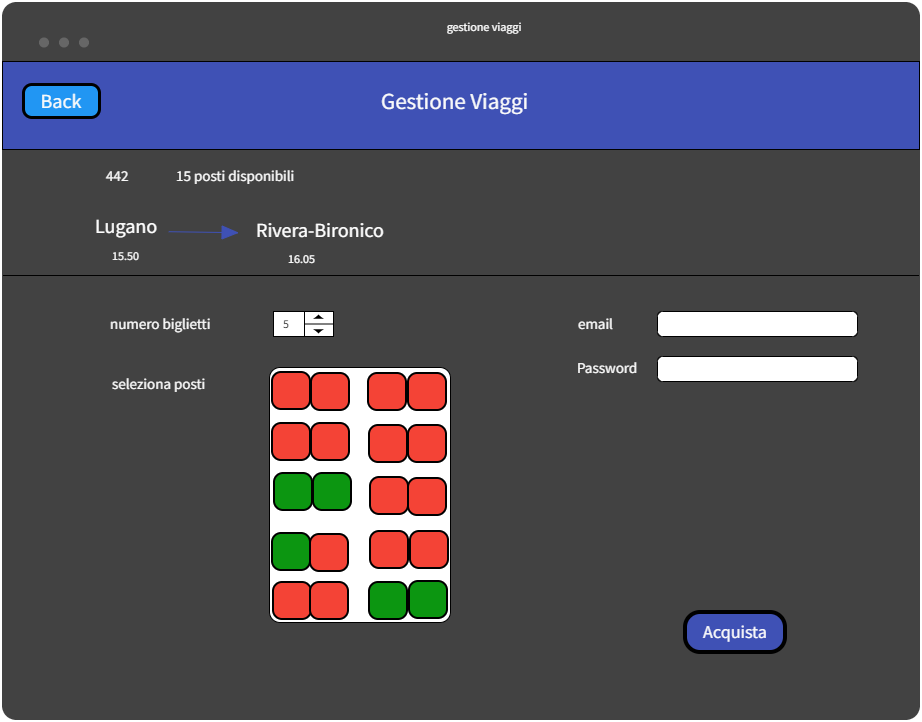
Pagina per cercare viaggi immettendo il luogo di partenza, il luogo di arrivo e la data ed orario in cui si vuole fare il viaggio. Poi premendo su cerca mostrerà tutti i viaggi previsti lungo quelle due località con il numero della linea, il numero di posti disponibili e l’orario in cui parte ed arriva a destinazione.  
Premendo sopra le tratte porterà ad una pagina con sui tutte le informazioni della tratta.

### Pagina mostra tratta



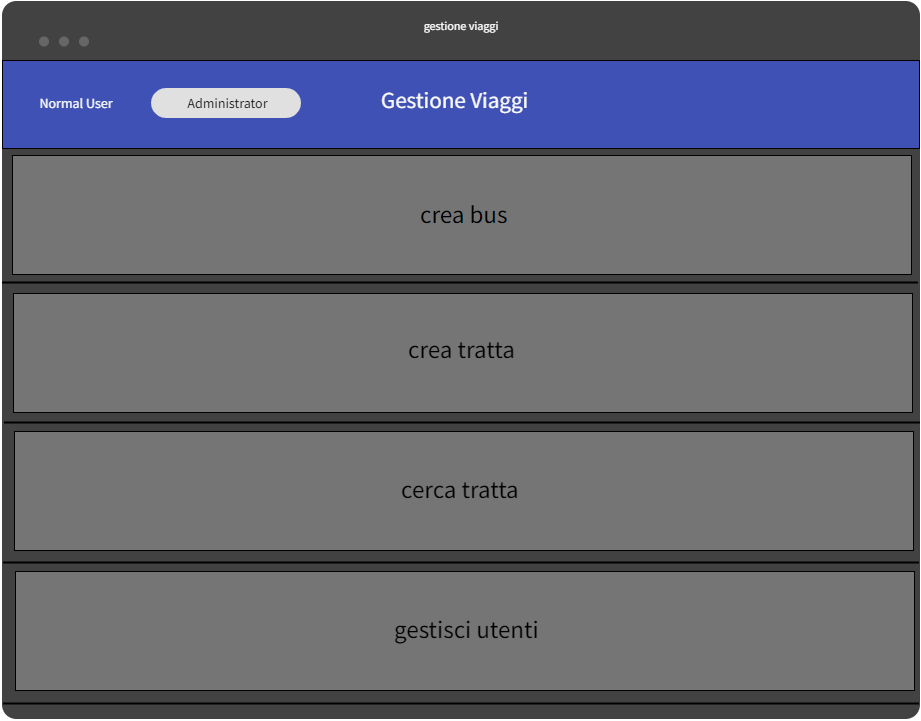
Pagina per mostra tutte le tappe in cui passa la tratta con l’orario in cui ci arriva e su in alto il numero della tratta ed il numero di posti disponibili. Se si è loggati nel proprio account allora apparirà anche un bottone acquista a destra della tratta che premendoci sopra porta alla pagina per acquistare biglietti per la tratta.

### Pagina acquista biglietti



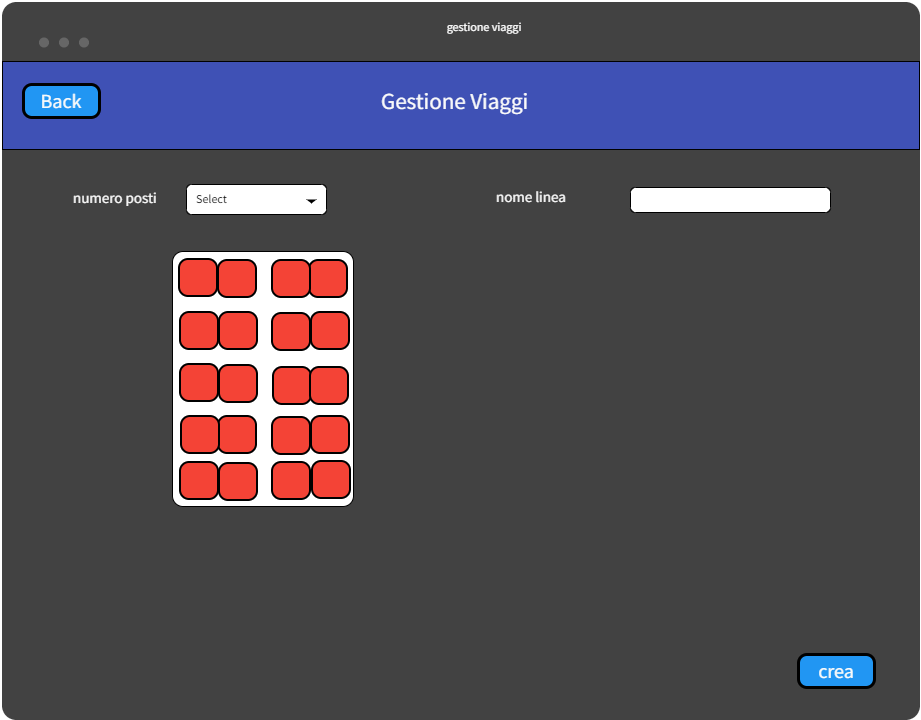
Pagina per acquistare biglietti accessibile solo se si è loggati nel sito. In alto viene mostrato il numero della tratta con i posti disponibili di fianco con sotto la località e l’orario di partenza con la località e l’orario di arrivo.  
Sotto ci sarà una tabella con su tutti i posti del bus in cui quelli colorati di rosso sono già stati riservati da altri utenti e quelli verdi sono riservabili premendo sopra a quelli che si vogliono riservare e poi premendo sopra acquista vengono acquistati.

### Pagina home amministratore



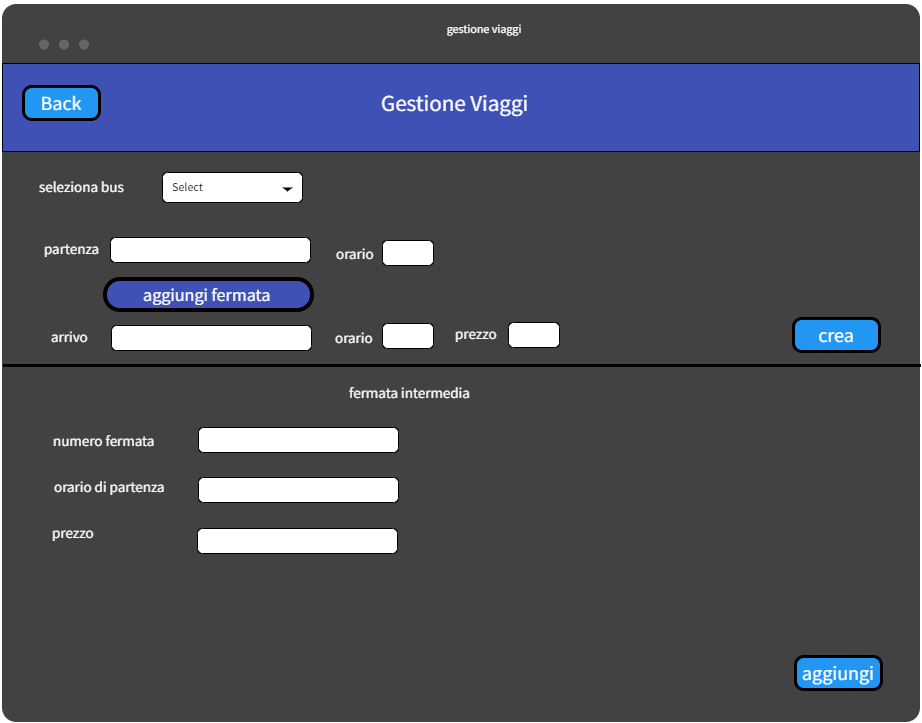
Pagina di home dopo un login di un account amministratore in cui ci sono 4 bottoni per portare alle pagine di creazione di bus e tratte, per cercare le tratte come per un utente normale, e la pagina per gestire gli utenti.

### Pagina crea bus



Pagina per creare un bus scegliendo tra 3 grandezze ossia, grande: 60 posti, medio: 40 posti e piccolo: 20 posti e dando un nome alla linea. Premendo su crea il bus verrà aggiunto al database.

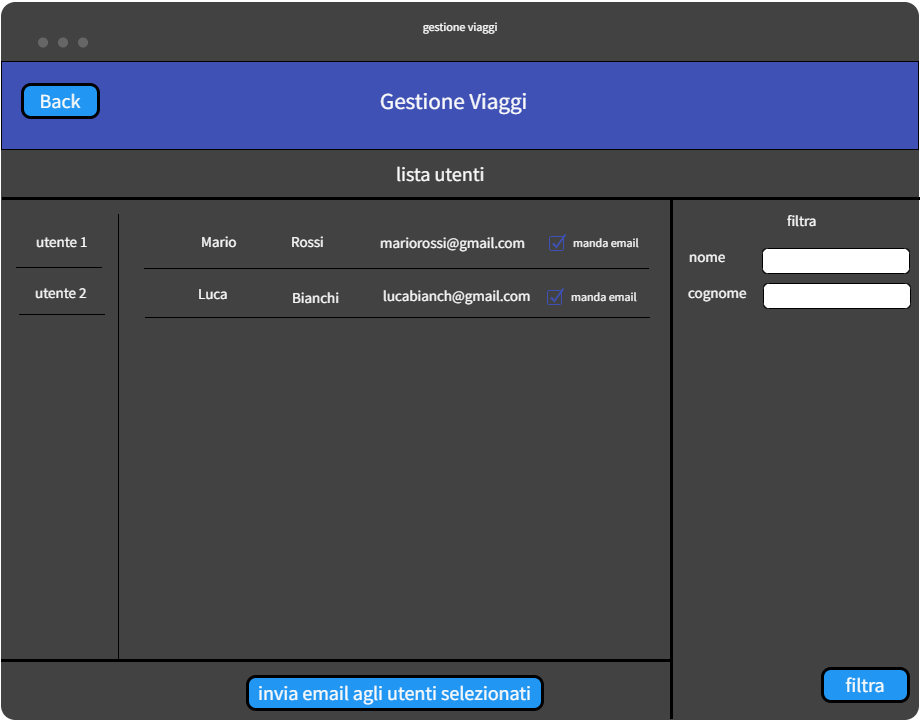
### Pagina crea tratta



Pagina per creare una tratta. Nel primo select si dovrà poter selezionare quale bus porterà lungo la tratta tra tutti i bus non occupati del database. Poi si dovrà aggiungere il nome della località di partenza assieme all’orario di partenza e si dovrà fare la stessa cosa per l’arrivo. Nella parte sotto della finestra si potrà aggiungere una fermata intermedia immettendo il nome della fermata, l’orario in cui parte dalla fermata ed il prezzo per arrivarci. Poi si premendo su aggiungi la fermata verrà inserita tra la partenza e la destinazione.

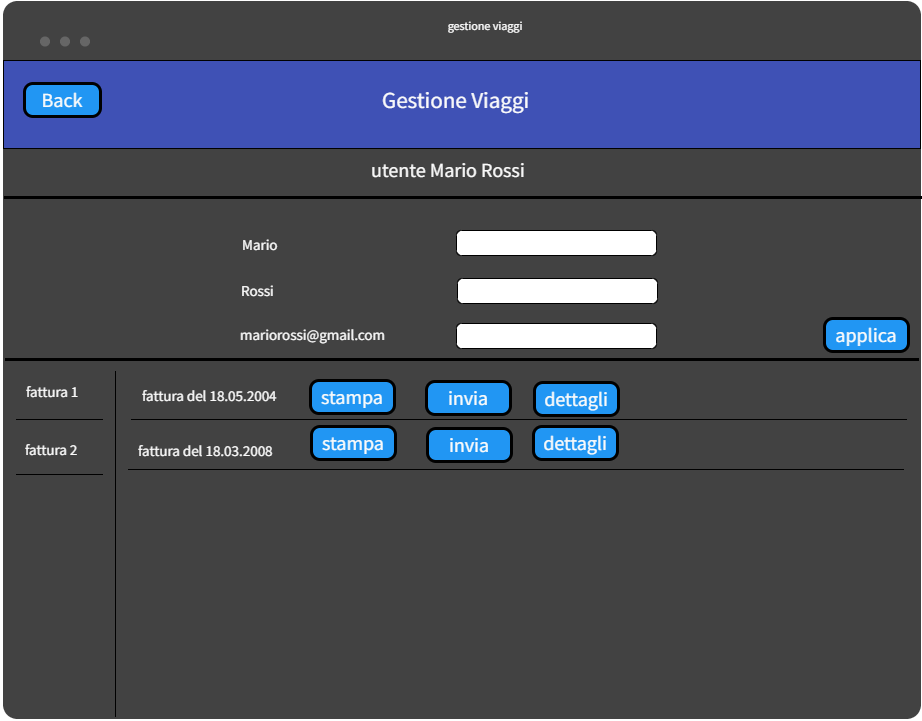
Premendo su crea la tratta verrà creata ed aggiunta al database.

### Pagina gestione utenti



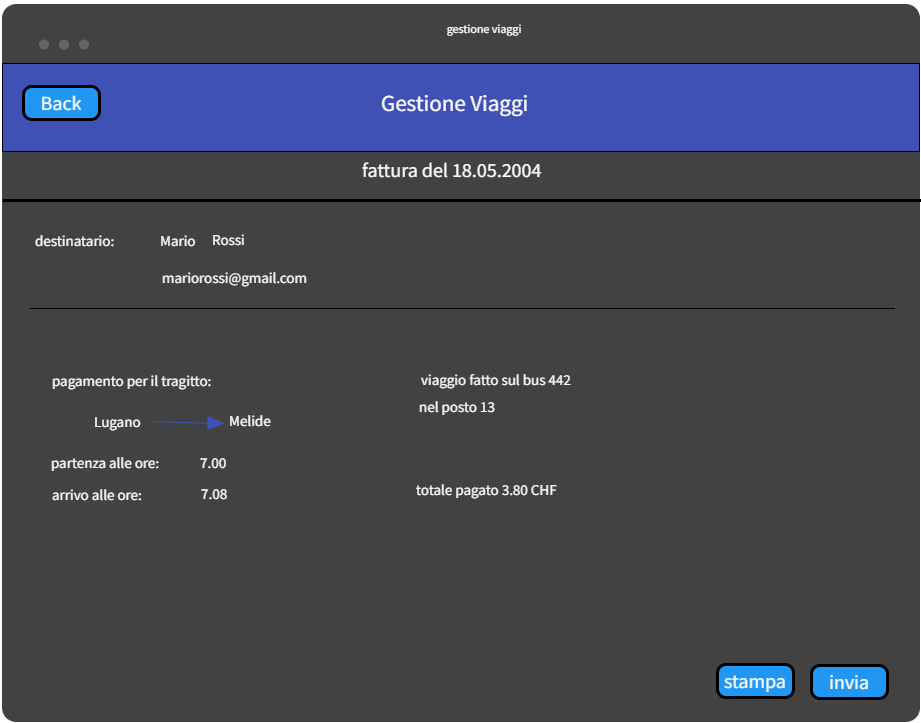
Pagina che mostra tutti gli utenti esistenti nel database con il loro nome, cognome, e-mail ed un visto per potergli mandare una mail. In fondo alla pagina ci sarà un bottone per poter mandare una mail a tutti gli utenti selezionati. A destra della pagina c’è un input per poter inserire un nome e/o cognome per cui filtrare la lista di utenti per poter trovare gli utenti desiderati più facilmente. Premendo sopra ad un utente porterà alla pagina dettagli utente dell’utente premuto.

### Pagina dettagli utente



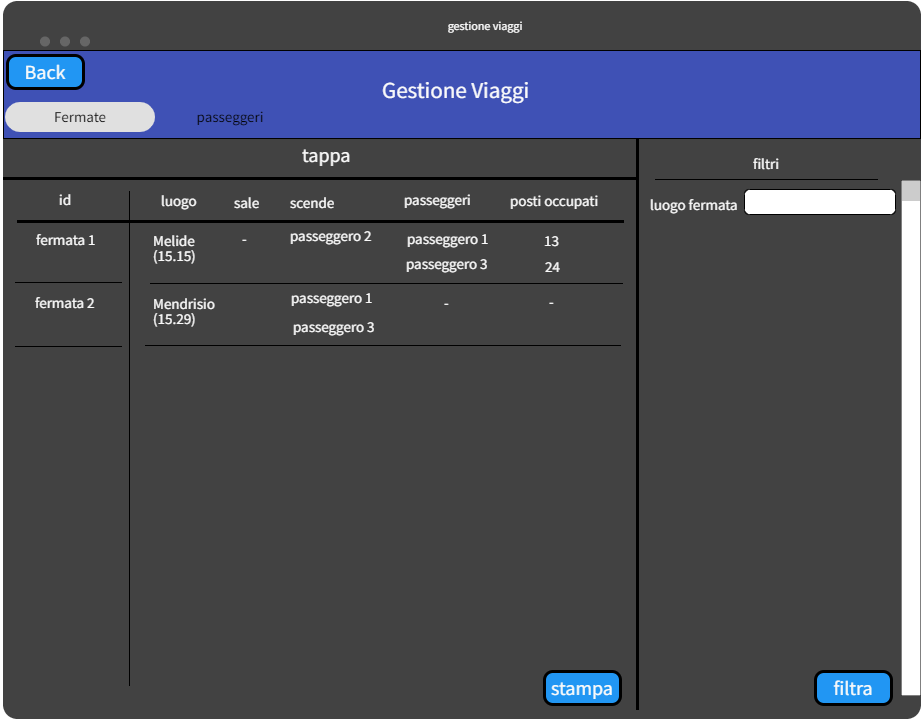
Pagina sulla quale vengono mostrati nome, cognome ed e-mail di un utente con di fianco ad ognuno un input per poterli modificare dopo aver premuto sul bottone applica.  
Sotto ci saranno le fatture di tutti i biglietti acquistati dall’utente con di fianco un bottone per stampare la fattura, inviarla ed uno che porta alla pagina per visualizzare i dettagli di una fattura.

### Pagina fattura



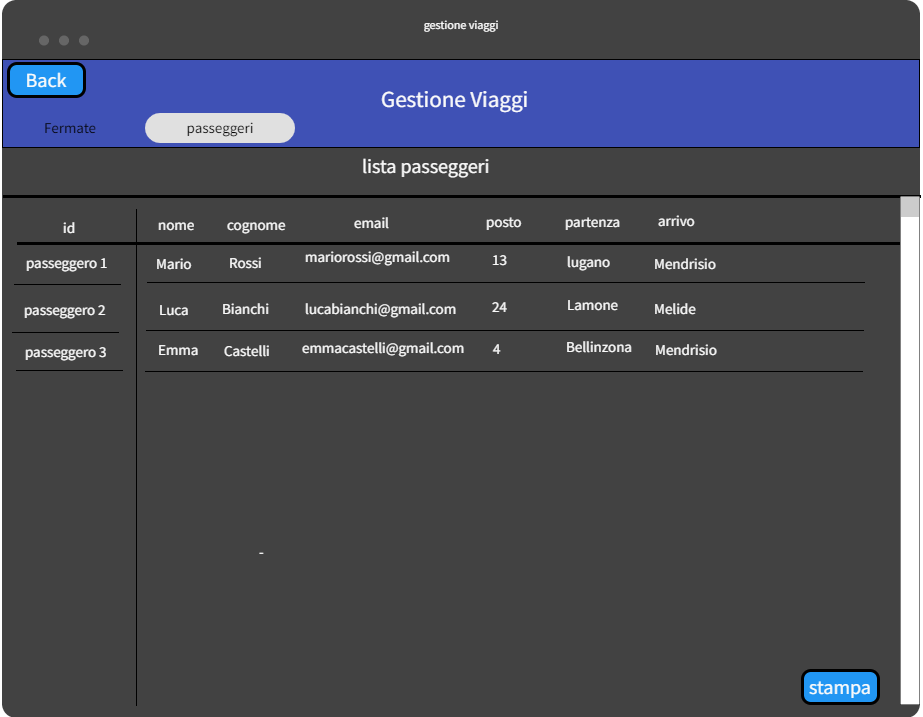
Pagina che mostra i dettagli di una fattura. In cima ci sarà la data della fattura con sotto i dettagli dell’utente che ha comprato il biglietto. Ancora sotto ci saranno le informazioni sul viaggio con gli orari e nomi delle località di partenza ed arrivo, il numero del bus ed il posto, ed il totale da pagare.  
In fondo alla pagina ci sarà un bottone per poter stampare la fattura ed un altro per poterla inviare.

### Pagina fermate



Pagina che mostra tutte le fermate di un viaggio con di fianco ad ogni fermata il nome e l’orario in cui si parte, che passeggeri salgono e quali scendono, e tutti i passeggeri a bordo con il numero del loro posto.  
A destra della pagina si potrà filtrare le fermate per il nome del luogo della fermata.  
In cima alla pagina si potrà cambiare pagina dalla visualizzazione delle fermate di un viaggio alla visualizzazione dei passeggeri di un viaggio.

### Pagina passeggeri



Pagina che mostra tutti i passeggeri di un viaggio con di fianco le loro informazioni tra cui: nome, cognome, e-mail, numero di posto occupato sul viaggio, località su cui salgono sul bus e località in cui scendono.  
In fondo a destra alla pagina ci sarà un bottone per poter stampare la lista dei passeggeri.

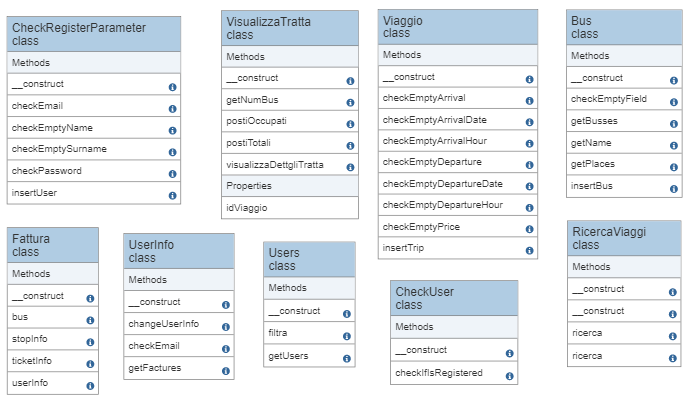
## Diagrammi UML

### Diagramma UML del controller

Diagramma UML del controller home.

### 

### Diagrammi UML delle classi nella cartella models.



# Implementazione

## Struttura applicativo

Abbiamo usato una struttura MVC in cui ogni pagina passa per una funzione nel controller home.php e da lì prende le view ed i model di cui ha bisogno per caricare correttamente la pagina e fare controlli sui dati che vengono passati.

La struttura delle cartelle che compongono il codice è la seguente:

├── Config

├── Controller

├── Libs

├── Models

└── Views

├── Amministratore

├── CercaViaggi

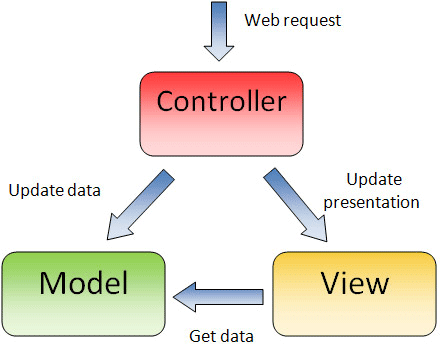
├── Footer

├── Header

├── UtenteLoggato

└── UtenteNonLoggato

La cartella “Controller” contiene il controller che risponde ai percorsi dati. La cartella “Libs” contiene il file che connette al database ed il file application. La cartella “Models” contiene tutte le classi oggetto. La cartella “Views” contiene tutte le interfacce del sito suddivise in varie cartelle a seconda di dove dovrebbero essere usate.



Nela struttura MVC ci sono 3 categorie:

* **Controller:** Contiene tutte le funzioni per caricare le pagine con le interfacce ed i dati corretti.
* **Model:** Classi che servono per leggere i dati dal database per potergli mostrare sulle interfacce ed anche per poter aggiungere dati al database.
* **View:** Le interfacce grafiche che vengono caricate sulla pagina.

## Database

Per il database abbiamo usato MySQL in cui abbiamo creato una tabella per ogni Entità del diagramma ER ed il sito si collega al database ogni volta che deve leggere o scrivere dati usando un file di connessione nella cartella /libs/ del progetto.

Il database è stato creato da workbench, esempio della creazione di una tabella:

CREATE TABLE IF NOT EXISTS bus(

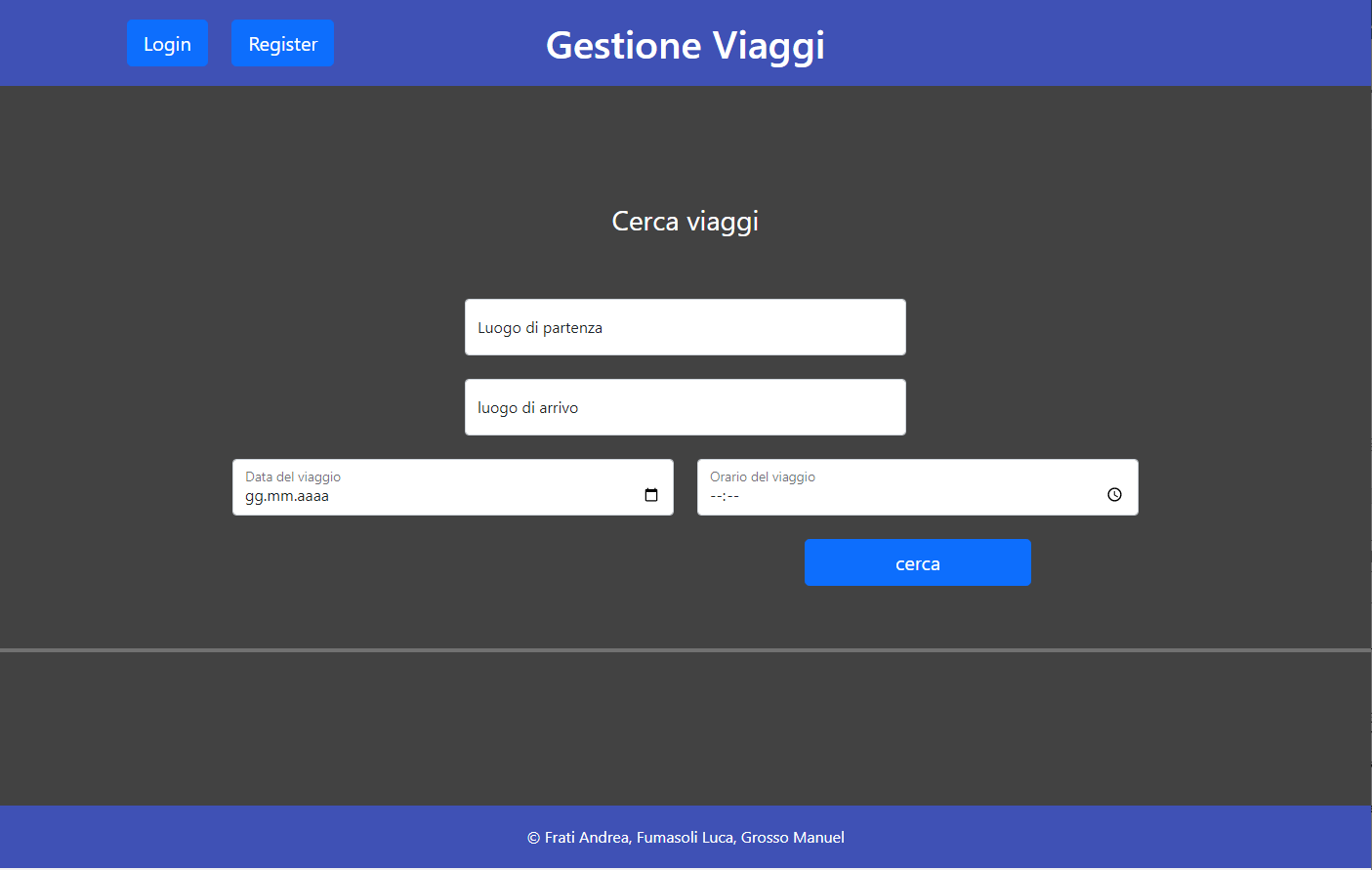
    id INT PRIMARY KEY AUTO\_INCREMENT,

    posti\_totali INT,

    numero\_bus INT

);

## Pagina di home



L’intera view per la ricerca dei viaggi è contenuta in un form di tipo POST che una volta premuto sul bottone cerca fa partire il metodo viaggi nel controller, nella pagina ci saranno 2 input di tipo testo per inserire il luogo di partenza e di arrivo, un input di tipo date per la data del viaggio ed un input di tipo time per l’orario del viaggio. Per mostrare cosa va messo nell’input abbiamo usato dei label che appaiono all’interno dell’input.  
Una volta immessi i dati ed avuto premuto su cerca appariranno le tratte esistenti che soddisfano i criteri di ricerca.



Questo viene fatto passando i valori dell’input con il form al metodo viaggi:

public function viaggi()

    {

        session\_start();

        if ($\_SESSION['isRegistered'] == false) {

           require\_once "application/views/header/headerLoginRegisterView.php";

        }else{

            require\_once "application/views/header/loggedUserHeader.php";

        }

Come prima cosa avvia una sessione e controlla se chi sta cercando il viaggio è un utente loggato o no per far apparire l’header giusto.

        require\_once "application/views/cercaViaggi/ricercaView.php";

        require\_once "application/models/ricercaViaggi.php";

Poi prende la view della parte superiore della pagina e la classe model ricerca viaggi.

        if (!isset($\_SESSION['partenza']) || !isset($\_SESSION['arrivo']) || !isset($\_SESSION['data']) || !isset($\_SESSION['orario']) ||isset($\_POST['luogoPartenza']) && $\_SESSION['partenza'] != $\_POST['luogoPartenza'] || isset($\_POST['luogoArrivo']) && $\_SESSION['arrivo'] != $\_POST['luogoArrivo'] || isset($\_POST['dataViaggio']) && $\_SESSION['data'] != $\_POST['dataViaggio'] ||  isset($\_POST['orarioViaggio']) && $\_SESSION['orario'] != $\_POST['orarioViaggio'] ) {

            $\_SESSION['partenza'] = $\_POST['luogoPartenza'];

            $\_SESSION['arrivo'] = $\_POST['luogoArrivo'];

            $\_SESSION['data'] = $\_POST['dataViaggio'];

            $\_SESSION['orario'] = $\_POST['orarioViaggio'];

        }

Fa vari controlli sulla correttezza dei parametri e poi setta delle variabili di sessione con i valori immessi.

        $partenza = $\_SESSION['partenza'];

        $arrivo = $\_SESSION['arrivo'];

        $data = $\_SESSION['data'];

        $orario = $\_SESSION['orario'];

Setta questi valori anche a delle variabili.

        if($data == "" && $orario == ""){

            $data = date("Y-m-d");

            $orario = date("H:i:s");

        }else if($orario == ""){

            $orario = date("00:00:00");

        }else if($data == ""){

            $data = date("Y-m-d");

        }

Poi controlla se la data o l’orario sono vuoti, e in quel caso gli setta i valori di oggi.

        echo "<script>console.log('PHP ".$data.$orario."' )</script>";

        $ricercaViaggi = new RicercaViaggi($partenza, $arrivo, $data, $orario);

        $result = $ricercaViaggi->ricerca();

crea un oggetto ricercaViaggi inserendo tutti i dati inviati come parametri e poi fa una ricerca nel database dei viaggi secondo questi parametri.

        require\_once "application/views/utenteNonLoggato/viaggiView.php";

        require\_once "application/views/footer/footerView.php";

    }

Infine prende la view che mostrerà tutti i viaggi trovati facendo un ciclo per tutti i risultati e stampando tutti i dati dell’array result.

        <a href="<?php

            if($\_SESSION['isRegistered'] == false){

                echo URL . 'home/tratta/';

            }else{

                echo URL . 'home/trattaLoggato/';

            }

            echo ($result[$i]['id\_viaggio'])?>" style="text-decoration: none;">

Premendo sopra la tratta, viene eseguito un controllo per vedere se l’utente è loggato o no e poi lo porta alla pagina dove mostra le informazioni sulla tratta.

## Pagina mostra tratta



In questa pagina il controller crea un oggetto visualizzaTratta passando come parametro al costruttore l’id del viaggio per sapere quale tratta prendere nel database e poi prende i dettagli sulla tratta.

        $visualizzaTratta = new VisualizzaTratta($idViaggio);

        $result = $visualizzaTratta->visualizzaDettgliTratta();

Poi nella view usando questo oggetto viene stampato il numero del bus, i posti disponibili, la partenza con l’orario, con un ciclo vengono stampate tutte le fermate intermedie con il nome della località e l’orario di partenza, ed infine la località d’arrivo con il proprio orario di arrivo.

Se si è loggati nel sito allora apparirà un bottone con su acquista sotto alla tratta che porterà ad una pagina per acquistare un biglietto passando in un form le informazioni della tratta.



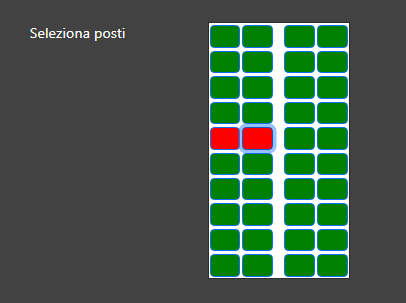
## Pagina acquista biglietto



Nella funzione del controller di questa pagina viene creato un oggetto di tipo bus per trovare le dimensioni del bus della tratta.

In cima a questa pagina appaiono le informazioni della tratta passate dalla pagina precedente ossia il numero della linea, il numero di posti e la partenza e l’arrivo con le proprie date e orari.

Sotto a questo c’è una griglia con su tutti i posti del bus sulla quale si può premere sopra i posti che si vuole riservare. Quando si preme sopra i posti diventano rossi tramite un metodo javascript.

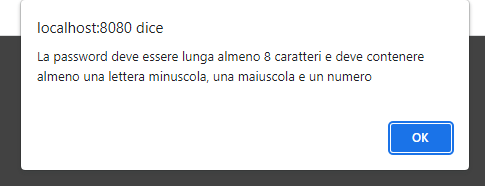


Poi premendo sul bottone acquista parte una funzione javascript che prende gli id di tutti i posti selezionati e gli passa in un form POST ad una funzione nel controller. Seguentemente questi id passati vengono acquistati ed aggiunti al database come posti riservati.

## Pagina registrazione



In questa pagina ci sono 5 input contenuti nel form: 2 input di tipo text per il nome ed il cognome, un input di tipo email per l’e-mail, e 2 input di tipo password per la password e la conferma della password. Poi quando si preme su registra tutti questi parametri vengono controllati e se sono corretti viene aggiunto l’account al database.  
Se non sono corretti o vuoti apparirà un pop-up che segnala gli errori commessi e poi riporterà alla pagina di registrazione dove gli input dei parametri sbagliati appaiono con attorno una cornice rossa.





Premendo invece sul bottone accedi sotto ti porta alla pagina di login.

## Pagina di login



C’è un input di tipo email per l’email ed uno di tipo password per la password all’interno di un form, quando si preme su accedi viene fatta partire la funzione tryToLogin che crea un oggetto CheckUser per controllare se i parametri sono sbagliati, se lo sono riporta alla stessa pagina ma con i contorni degli input sbagliati rossi come sulla pagina di registrazione, altrimenti viene controllato se l’account è di un utente normale o di un amministratore e lo porta alla home degli utenti o degli amministratori a dipendenza di questo.  
Premendo sul bottone registrati sotto ti porta alla pagina di registrazione.

## Pagina di home - amministratore



Su questa pagina ci sono 4 <h3> con su il nome delle pagine a cui ha accesso l’amministratore ed ognuno è circondato da un tag <a> che porta alla pagina titolata.

## Pagina di creazione bus



Pagina con su in alto un select per scegliere la grandezza del bus tra Piccolo, Medio e Grande, e quando si preme conferma parte il form che aggiorna la pagina con il valore della grandezza del bus appena scelto e stampa la tabella sotto in base a questo valore. Di fianco c’è un input di tipo text per il nome della linea del bus. Quando si preme su crea viene creato un oggetto di tipo bus e controlla se il parametro nome linea passato è valido, se non lo è riporta alla pagina con la cornice dell’input colorata di rosso. Se il parametro è corretto lo aggiunge al database.

## Pagina di creazione viaggi



Per la selezione del bus ………………… non ancora fatta………………

Per la località di partenza e di arrivo c’è un input di tipo testo e per le date e orari c’è un input di tipo datetime.  
Nella parte sotto della pagina ci sono gli stessi 2 input per la località e l’orario e per il prezzo c’è un input di tipo number. Quando si preme aggiungi fermata i dati della fermata vengono aggiunti tra la partenza e l’arrivo,

## Pagina gestione utenti



Il controller crea un oggetto users e usando il metodo getUsers salva tutti gli utenti in una variabile, poi quando carica la view vengono stampati tutti gli utenti contenuti nel database con un ciclo for sulla pagina.

In cima alla pagina ci sono 2 input di tipo testo per il nome ed il cognome. Quando si preme su filtra parte la funzione users che crea di nuovo un oggetto di tipo users e salva in una variabile tutti gli utenti, poi usa il metodo filtra prendendo i parametri passati dal form ed applicandogli sulla variabile contenente tutti gli utenti.

Poi ricarica la view stampando l’array di utenti contenente solo gli utenti desiderati.

Di fianco agli utenti c’è uno switch che si può premere se si vuole mandare una email all’utente, se si vuole mandare un’email a tutti gli utenti si può premere lo switch a destra del bottone filtra così selezionerà tutti gli utenti.  
Poi premendo sul bottone “manda email agli utenti selezionati” verrà inviata una mail ad ogni utente selezionato.

Ogni utente è contenuto in un div ed attorno ad ogni div c’è un tag a che quando viene premuto sopra porta alla pagina informazioni utente dell’utente su cui si ha premuto.

## Pagina informazioni utenti



In cima alla pagina viene scritto il nome dell’utente che viene passato dalla funzione precedente.

Lo stesso vale per il nome ed il cognome dentro i primi due input di tipo testo ed il terzo di tipo email.  
Per mostrare le fatture sotto viene creato un oggetto di tipo userInfo e poi vengono salvate le fatture in una variabile usando il metodo getFractures($email) che passando come parametro l’email trova tutte le fatture nel database corrispondenti a questo utente. Poi quando la view viene creata con un ciclo scorre tutte le fatture in un div con i loro dati e le circonda con un tag a che quando viene premuto sopra porta alla pagina delle specifiche della fattura premuta.

Il nome, cognome e la password sono contenute in degli input modificabili che quando vengono modificate e si preme sul bottone applica parte la funzione userInfoChanged($nome, $cognome, $email), crea un nuovo oggetto userInfo, prende i nuovi dati dell’utente modificati e controlla che i campi non siano vuoti.

Poi chiamando il metodo changeUserInfo($nome, $cognome,$newEmail,$email) vengono cambiati i dati dell’utente nel datbase con quelli nuovi. Infine viene ricaricata la pagina con le fatture come prima.

## Pagina fatture



Nel controller viene creato un oggetto di tipo fattura e poi vengono usati i metodi ticketInfo($id), userInfo($id), stopInfo($id) e bus($idViaggio) per ricavare le informazioni della fattura e poi quando viene chiamata la view viene stampata con tutte le informazioni al posto giusto.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-10 | **Nome:** | Registrazione di un utente non registrato |
| **Descrizione:** | Un utente non registrato ha la possibilità di registrarsi cliccando su “Register”.  Una volta cliccato si viene portati nella pagina di registrazione dove si inserisce nome, cognome, email, password e conferma password.  Se tutti I valori verranno messi correttamente si passerà alla pagina da utente loggato e si potranno acquistare biglietti. | | |
| **Prerequisiti:** | Avere un database dove salvare gli utenti gia registrati con la possibilità di aggiungere nuovi utenti. | | |
| **Procedura:** | 1. Cliccare sul bottone “Register”. 2. Inserire le proprie credenziali 3. Cliccare su “Register” 4. Controllare se c’è la possibilità di rieffettuare il login e di comprare dei biglietti | | |
| **Risultati attesi:** | Possibilità di rifare il login se ci si disconnette e di acquistare I biglietti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-03 | **Nome:** | Interfaccie diverse tra i livelli di utenti |
| **Descrizione:** | A dipendenza del tipo di utente con la quale si ha fatto il login verranno visualizzate delle interfaccie differenti. | | |
| **Prerequisiti:** | Poter registrare un utente di vario tipo nel database e avere tutte e tre le interfaccie per i vari utenti. | | |
| **Procedura:** | 1. Aprire la pagina da utente non registrato e controllare se viene mostrata la view corretta. 2. Loggarsi con un normale utente e controllare se viene mostrata la view corretta. 3. Effetturare il logout, fare login con un utente amministratore e controllare che viene mostrata la view corretta. | | |
| **Risultati attesi:** | Ad ogni tipo di utente verrà mostrata una view differente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-04 | **Nome:** | Inserire nel database un tragitto |
| **Descrizione:** | Un utente di tipo amministratore può creare nuovi tragitti. | | |
| **Prerequisiti:** | Avere il database funzionante con la possibilità di aggiungere tragitti e capire se l’utente loggato è un amministratore. | | |
| **Procedura:** | 1. Fare il login con un utente amministatore 2. Cliccare sul bottone aggiungi tratte 3. Mettere tutti i parametri richiesti per creare una tratta 4. Cliccare su aggiungi tratta 5. Controllare che la tratta è stata aggiunta correttamente | | |
| **Risultati attesi:** | Se cerchiamo la tratta appena aggiunta sarà presente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-05 | **Nome:** | Controllare e gestire i partecipanti dei viaggi |
| **Descrizione:** | Un utente di tipo amministratore può visualizzare tutti gli utenti che hanno acquistato un biglietto nella tratta selezionata e può vedere e modificare I dati da loro inseriti.. | | |
| **Prerequisiti:** | Avere il database funzionante, capire se l’utente loggato è un amministratore e se lo è ha la possibilità di aggiungere tragitti.  Poter creare utenti che possono acquistare I biglietti | | |
| **Procedura:** | 1. Fare il login con un utente amministatore 2. Ricercare una tratta 3. Cliccare sulla tratta desiderata 4. Verrà mostrata una lista dei partecipanti, selezionarne uno 5. Modificare I suoi dati. | | |
| **Risultati attesi:** | Controllare che una volt ache I ati sono stati modificati si salvano corretamente sul database e si salvano. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-06 | **Nome:** | Stampare lista dei partecipanti di un viaggio |
| **Descrizione:** | Un utente di tipo amministratore può visualizzare tutti gli utenti che hanno acquistato un biglietto nella tratta selezionata e può stamparne la lista. | | |
| **Prerequisiti:** | Avere il database funzionante, capire se l’utente loggato è un amministratore e se lo è ha la possibilità di aggiungere tragitti.  Poter creare utenti che possono acquistare i biglietti | | |
| **Procedura:** | 1. Fare il login con un utente amministatore 2. Ricercare una tratta 3. Cliccare sulla tratta desiderata 4. Verrà mostrata una lista dei partecipanti con un tasto stampa 5. Selezionare la stampante e stampare | | |
| **Risultati attesi:** | Una volta mandata la stampa controllare che venga fisicamente stampata la lista di partecianti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-07 | **Nome:** | Sistema filtrabile di mass mail |
| **Descrizione:** | Un utente di tipo amministratore può visualizzare tutti gli utenti dalla gestione utenti filtrarli e mandare una mail a tutti quelli selezionati.. | | |
| **Prerequisiti:** | Avere il database funzionante, capire se l’utente loggato è un amministratore.  Poter creare utenti normali che si salvano sul database. | | |
| **Procedura:** | 1. Fare il login con un utente amministatore 2. Cliccare su gestione utenti 3. Filtrare a piacimento gli utenti. 4. Cliccare su “Invia mail” | | |
| **Risultati attesi:** | Una volta mandata la mail controllare che sia arrivata a tutti gli utenti selezionati | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-08 | **Nome:** | Gestione dei pagamenti |
| **Descrizione:** | Un utente di tipo amministratore può visualizzare tutti gli utenti dalla gestione utenti filtrarli e selezionare un utente per vedere nel dettaglio i dati e i pagamenti | | |
| **Prerequisiti:** | Avere il database funzionante, capire se l’utente loggato è un amministratore.  Poter creare utenti normali che si salvano sul database e possono aquistare viaggi. | | |
| **Procedura:** | 1. Fare il login con un utente amministatore 2. Cliccare su gestione utenti 3. Filtrare a piacimento gli utenti. 4. Selezionare l’utente desiderato 5. Possibilità di vedere e modificare i pagamenti | | |
| **Risultati attesi:** | Una volta modificati controllare che vengono salvati correttamente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-09 | **Nome:** | Ricerca tragitto con qualsiasi tipo di utente |
| **Descrizione:** | Qualisiasi tipo di utente può ricercare e filtrare un viaggio. | | |
| **Prerequisiti:** | Avere il database funzionante, aver la possibilità di creare dei viaggi da un utente amministatore.  Avere tutti e tre i tipi di account | | |
| **Procedura:** | 1. Ricercare un viaggio 2. Filtrare i viaggi 3. Cliccare su “Cerca” 4. Ripetere con tutti i tipi di utente | | |
| **Risultati attesi:** | Con ogni tipo di utente una volta schiacciato cerca devono apparire I viaggi correti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-00?  REQ-011 | **Nome:** | Log-in al sito |
| **Descrizione:** | Un utente dovrà potersi loggare al sito. | | |
| **Prerequisiti:** | Avere il database funzionante, con un utente creato. | | |
| **Procedura:** | 1. Andare sulla pagina di login del sito 2. Immettere l’email e la password 3. Premere il bottone login | | |
| **Risultati attesi:** | Una volta premuto login dovrà portarti alla pagina di home del sito e si dovrà vedere che si è loggati sull’icona del profilo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-00?  REQ-012 | **Nome:** | Prenotazione biglietto |
| **Descrizione:** | Un utente dovrà potersi loggare al sito. | | |
| **Prerequisiti:** | Avere il database funzionante, con un utente creato. | | |
| **Procedura:** | 1. Fare login sul sito 2. Premere su una tratta 3. Premere sul bottone acquista 4. Selezionare quanti biglietti si vogliono riservare 5. Premere sopra lo schema quali posti liberi si vogliono riservare 6. Premere effettua pagamento | | |
| **Risultati attesi:** | Si dovrà ricevere una mail di conferma dell’acquisto del biglietto con le informazioni del biglietto. | | |

## 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.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Si potrebbero implementare i requisiti che non siamo riusciti ad implementare in tempo, ...

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

### Luca Fumasoli

Durante questo progetto ho imparato ad usare meglio git, ho capito meglio come funziona la struttura MVC e come si collega un database MySQL ad un sito.

# Glossario

|  |  |
| --- | --- |
| **Nome** | **Definizione** |
| HTML | HyperText Markup Language, è un linguaggio di markup. |
| Javascript | Javascript è un linguaggio di programmazione web per gli eventi lato client. |
| CSS | Cascading Style Sheets, serve a stilizzare l’interfaccia delle pagine. |
| PHP | Hypertext Preprocessor, è un linguaggio di scripting interpretato. |
| UML | Unified Modeling Language, linguaggio di modellazione spesso usato per rappresentare classi. |
| ER | Modello entità relazione, è utilizzato per progettare e rappresentare le relazioni tra i dati. |
| MySQL | Software di gestione di database. |

# Sitografia

<https://www.php.net/manual/en/function.substr.php>, 24.02.22

<https://www.html.it/pag/54075/gestire-date-e-orari/>, gestire data e orari, 24.02.22

<https://getbootstrap.com/docs/>, 17.02.2022

<https://developer.mozilla.org/>, 17.02.2022

<https://stackoverflow.com/questions/585945/how-to-align-content-of-a-div-to-the-bottom>, 17.02.2022

<https://howtodoinjava.com/spring-mvc/spring-mvc-hello-world-example/>, imagine MVC, 05.05.2022

<https://cloud.smartdraw.com/>, diagrammi UML, 05.05.2023