

Documentazione Progetto SWA

Nome Progetto: Gestione Aule

Autori:

Nome	Cognome	Matricola	Ruolo
Denis	Ciammaricone	271778	Sviluppo back-end
Stefano	Bavota	258821	Sviluppo front-end
Lorenzo	Bosica	267240	Database e documentazione

Dipendenze software:

-Gestione dipendenze tramite Composer

Prerequisiti:

PHP versione 8 o superiore

Installazione:

<https://getcomposer.org/doc/00-intro.md>

Descrizione file composer.json:

Il file **composer.json** è un file di configurazione essenziale utilizzato da Composer, uno strumento di gestione delle dipendenze per progetti PHP. Questo file contiene informazioni fondamentali sul nostro progetto e definisce le librerie esterne (o pacchetti) richiesti dall'applicazione, oltre a specificare varie configurazioni.

Struttura del file

Il file **composer.json** ha una struttura chiara e ben definita. Alcuni dei campi principali presenti nel file includono:

- **name**: Il nome univoco del nostro progetto o pacchetto.
- **type**: Il tipo del progetto o pacchetto (es. "project" per progetti generici).
- **description**: Una breve descrizione del nostro progetto.
- **keywords**: Parole chiave associate al nostro progetto per facilitarne la ricerca.
- **license**: La licenza con cui il nostro progetto è distribuito (es. "MIT").
- **require**: Elenco delle dipendenze richieste dal nostro progetto.
- **require-dev**: Elenco delle dipendenze richieste solo durante lo sviluppo e i test del progetto.
- **autoload**: Configurazione per l'autoloading delle classi del nostro progetto.
- **scripts**: Definizione di script personalizzati eseguiti in vari eventi di Composer.
- Altre configurazioni opzionali come **extra**, **config**, ecc.

Dipendenze richieste

La sezione **require** del file **composer.json** elenca le librerie esterne richieste dal nostro progetto. Queste dipendenze includono le versioni minime delle librerie di

Laravel che il nostro progetto utilizza, oltre ad altre librerie esterne come "guzzlehttp/guzzle" per la gestione delle richieste HTTP.

Dipendenze per lo sviluppo

La sezione **require-dev** elenca le dipendenze richieste solo durante lo sviluppo e i test del progetto. Queste librerie includono strumenti per il testing come "phpunit/phpunit" e "mockery/mockery", insieme a altre utilità come "fakerphp/faker" per generare dati falsi per i test.

Autoloading delle classi

La sezione **autoload** definisce come le classi del nostro progetto vengono caricate automaticamente. È specificato il percorso delle classi per il namespace "App" e per le factory e seeders del database.

Script personalizzati

Nella sezione **scripts**, definiamo script personalizzati che vengono eseguiti in vari eventi di Composer. Ad esempio, dopo l'aggiornamento delle dipendenze, eseguiamo l'Artisan Command **vendor:publish --tag=laravel-assets** per pubblicare le risorse di Laravel.

Configurazione di Composer

La sezione **config** contiene alcune configurazioni di Composer per ottimizzare l'autoloading, ordinare i pacchetti e abilitare alcuni plugin utilizzati nel progetto.

Installazione dipendenze:

Per installare le dipendenze definite all'interno del nostro progetto eseguire il comando **composer install**.

Descrizione file composer.lock:

Il file **composer.lock** è una componente essenziale per garantire la stabilità, la coerenza e la riproducibilità delle dipendenze del progetto.

Il file composer.lock contiene le seguenti informazioni:

- Elenco di tutte le librerie esterne installate nelle loro versioni esatte per garantire un corretto funzionamento del progetto .

-Docker

Docker è una tecnologia di containerizzazione che ci permette di creare ambienti virtualizzati, chiamati container, per eseguire le nostre applicazioni in modo isolato.

-Installazione

<https://www.docker.com/products/docker-desktop/>

Abbiamo scelto Docker per i seguenti motivi:

-Portabilità

Tramite Docker possiamo creare un pacchetto con all'interno tutta l'applicazione (applicazione e tutte le sue dipendenze) e distribuirlo

come un'unica unità indipendentemente dalla piattaforma su cui viene distribuito. Docker ci consente di specificare le versioni esatte delle dipendenze dell'applicazione all'interno del Dockerfile o tramite il file **docker-compose.yml**. Questo assicura la riproducibilità dell'ambiente di esecuzione, garantendo che tutti gli ambienti utilizzino le stesse versioni delle librerie e dei servizi.

-Isolamento

I container Docker operano in modo isolato dagli altri processi del sistema. Questo significa che possiamo eseguire più container contemporaneamente senza interferenze tra loro. Ogni container ha il proprio filesystem, rete e processi, garantendo una maggiore sicurezza e stabilità dell'applicazione.

Tecnologie/Librerie/Framework Client:

-HTML

HTML (HyperText Markup Language) è il linguaggio di markup standard utilizzato per strutturare il contenuto delle pagine web. Nella nostra applicazione, utilizziamo HTML per definire la struttura dei componenti e dei template dell'interfaccia utente.

-CSS

CSS (Cascading Style Sheets) è il linguaggio utilizzato per definire lo stile e la presentazione delle pagine web. Nella nostra applicazione, utilizziamo CSS per gestire l'aspetto grafico dell'interfaccia utente.

-JavaScript

JavaScript è un linguaggio di programmazione client-side ampiamente utilizzato per aggiungere interattività e funzionalità dinamiche alle pagine web.

-Angular

Angular è un framework di sviluppo front-end sviluppato da Google che facilita la creazione di applicazioni web complesse e dinamiche. Includiamo Angular nel nostro progetto per sfruttare le sue funzionalità avanzate, come il data binding bidirezionale, la gestione delle route, l'iniezione delle dipendenze. Angular ci consente di organizzare il nostro codice in modo modulare e mantenibile, facilitando la realizzazione di applicazioni scalabili.

-Full-calendar

Full-calendar è una libreria JavaScript per la gestione delle date che estende le funzionalità del calendario. Utilizziamo Full-calendar per visualizzare, selezionare e gestire gli eventi e le date all'interno della nostra applicazione. La libreria offre una vasta gamma di funzioni, come la visualizzazione di diversi tipi di calendari, il supporto per gli eventi ricorrenti e altro ancora.

Tecnologie/Librerie/Framework Server:

-PHP

PHP è un linguaggio di scripting server-side ampiamente utilizzato per lo sviluppo web. Nella nostra applicazione, abbiamo utilizzato PHP per implementare la logica del back-end, elaborare le richieste HTTP, accedere al database e fornire le risposte ai client.

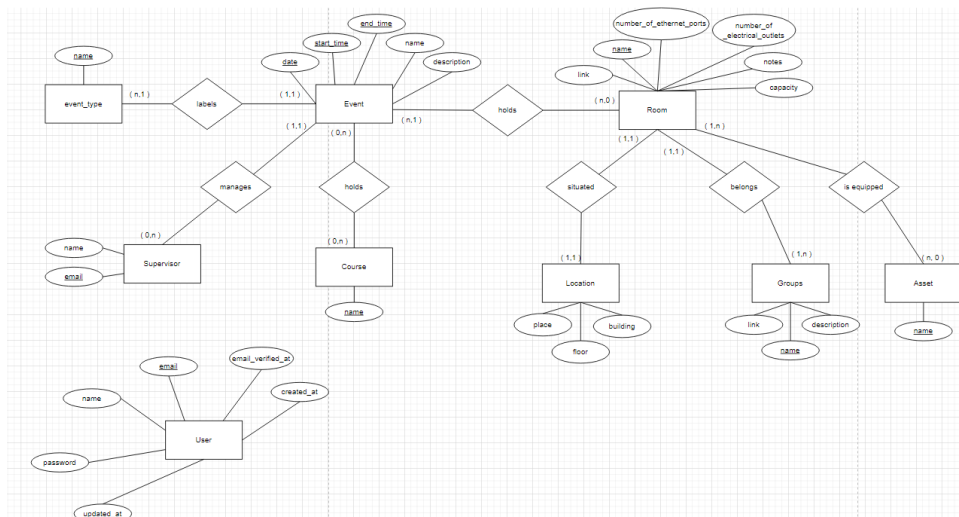
-Laravel

Laravel è un framework PHP open-source che semplifica notevolmente lo sviluppo di applicazioni web. Abbiamo scelto Laravel come framework per il nostro progetto poiché offre una vasta gamma di funzionalità, tra cui il sistema di routing, gestione delle sessioni. Utilizziamo Laravel per strutturare il nostro codice in modo modulare e mantenibile, consentendo una rapida implementazione delle funzionalità.

-Mysql

MySQL è un sistema di gestione di database relazionali (RDBMS) ampiamente utilizzato per la memorizzazione e la gestione dei dati. Nel nostro progetto, utilizziamo MySQL per creare e gestire il database dell'applicazione. Utilizziamo l'ORM eloquent di Laravel per interagire con il database in modo semplice e sicuro.

-Modello E/R DB



Descrizione Database

Tabelle:

Event(date, start_time, end_time, name, description)

Room(name, number_of_ethernet_ports,
number_of_electrical_outlets, notes, capacity, link)

Supervisor(email, name)

Course(name)

Location (place, floor, building)

Group(name, link, description)

Asset(name)

Event_type(name)

User(email, email_verified_at, created_at, name, password,
updated_at)

Dopo aver strutturato il modello del database abbiamo scelto di partire prima con una implementazione MySQL, per testare l'interazione delle relazioni fra le varie tabelle e i funzionamenti delle query, una volta verificato il tutto abbiamo iniziato lo sviluppo vero e proprio in Laravel (in allegato insieme alla documentazione il file MySQL, l'implementazione in Laravel differisce leggermente dalla prima implementazione in MySQL).

Funzionalità Realizzate:

1. Login/logout con username e password (per gli amministratori).
2. Esportazione e importazione CSV configurazione aule non funziona solo dal front-end.
3. Inserimento di una nuova aula.
4. Assegnazione di un'aula a un gruppo.
5. Lettura delle informazioni di base relative a un'aula.
6. Lista delle attrezzature presenti in un'aula.
7. Inserimento di un nuovo evento.
8. Modifica di un evento.
9. Lettura delle informazioni su un evento.
10. Lista degli eventi associati a una specifica aula in una determinata settimana.
11. Lista degli eventi attuali e quelli delle prossime tre ore.
12. Creazione, visualizzazione, modifica eventi
13. Creazione, visualizzazione, modifica gruppi
14. Creazione, visualizzazione, modifica classi
15. La cancellazione implementata lato back-end ma non lato front-end
16. L'export degli eventi la query è stata implementata e funziona ma non funziona quando richiamata dal front end

Funzionalità Non Realizzate:

1. La cancellazione degli oggetti non è stata implementata per motivi di tempo
2. La formattazione degli orari con scarti di 15 minuti
3. Importazione CSV configurazione aule
4. La pianificazione di eventi giornalieri, settimanali, ecc...

Specifiche delle API:

Per una visione migliore andare su <https://editor.swagger.io/> ed incollare il codice del file.yaml lasciato in allegato al progetto.

Esempi JSON:

Esempio JSON Entità gruppo
Data Consegna Progetto:
25/07/2023

```
[
  {
    "id": 0,
    "name": "string",
    "description": "string",
    "link": "string"
  }
]
```

Esempio JSON Entità Classe

```
[
  {
    "id": 0,
    "name": "string",
    "capacity": 0,
    "description": "string",
    "link": "string",
    "electricalOutlets": 0,
    "ethernetPorts": 0,
    "supervisor_id": 0,
    "group_id": 0,
    "location_id": 0
  }
]
```

Esempio Entità Evento

```
[
  {
    "id": 0,
    "name": "string",
    "description": "string",
    "date": "string",
    "start_date": "string",
    "end_date": "string",
    "typology_id": 0,
    "supervisor_id": 0,
    "room_id": 0,
    "room": {
      "id": 0,
      "name": "string",
      "capacity": 0,
      "description": "string",
      "link": "string",
      "electricalOutlets": 0,
      "ethernetPorts": 0,
      "supervisor_id": 0,
      "group_id": 0,
      "location_id": 0
    },
  },
]
```

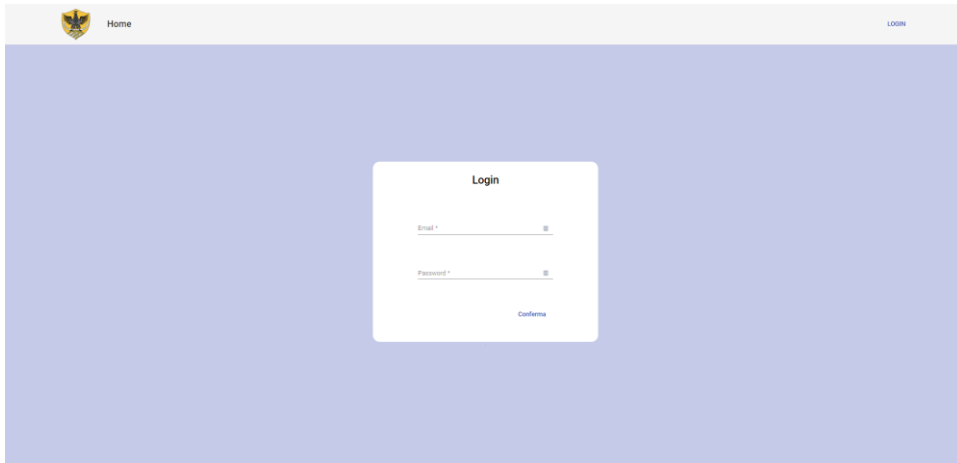
Lista Browser Compatibili:

- Google Chrome
- Microsoft Edge
- Opera
- Mozilla Firefox
- Brave
- Vivaldi

Data Consegna Progetto:
25/07/2023

Struttura del sito:

Pagina login

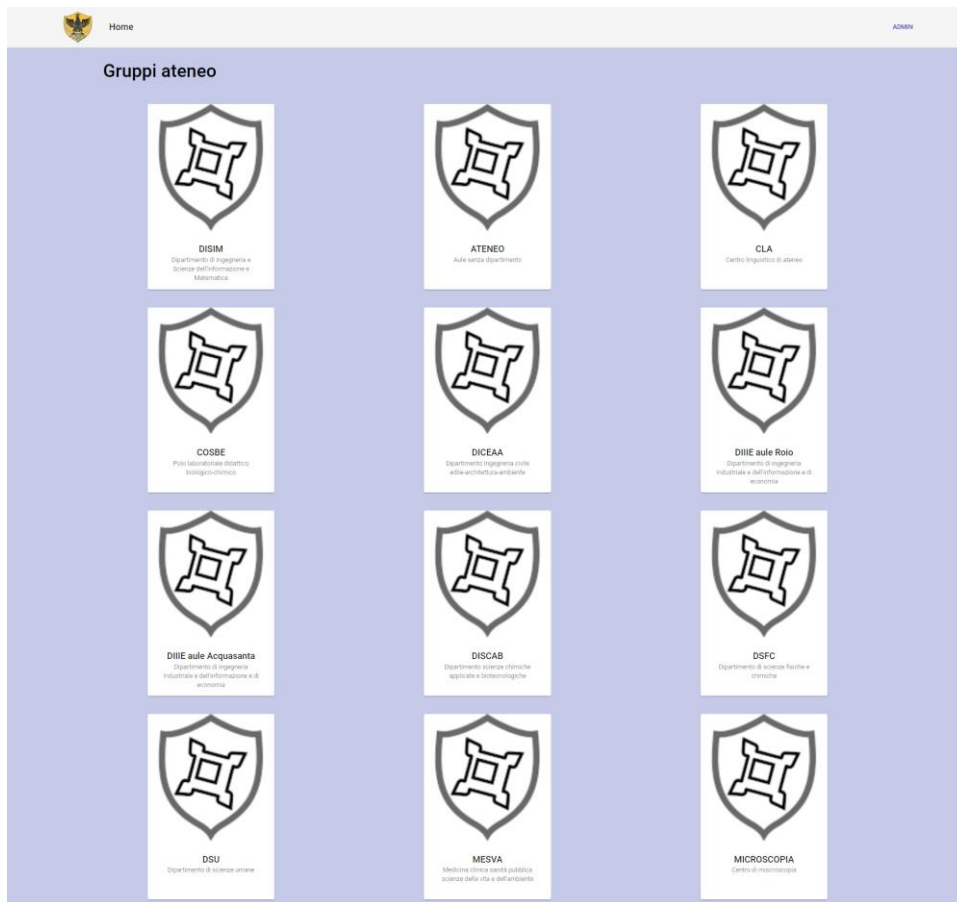


Nella pagina del login gli amministratori possono effettuare l'accesso tramite l'inserimento di e-mail e password.

Questa pagina sarà accessibile da tutto il sito se non si è autenticati tramite il tasto in alto a destra "LOGIN" se si è già autenticati il tasto verrà sostituito con il tasto "ADMIN".

In alto a sinistra invece c'è il tasto home che porta sulla pagina principale del sito.

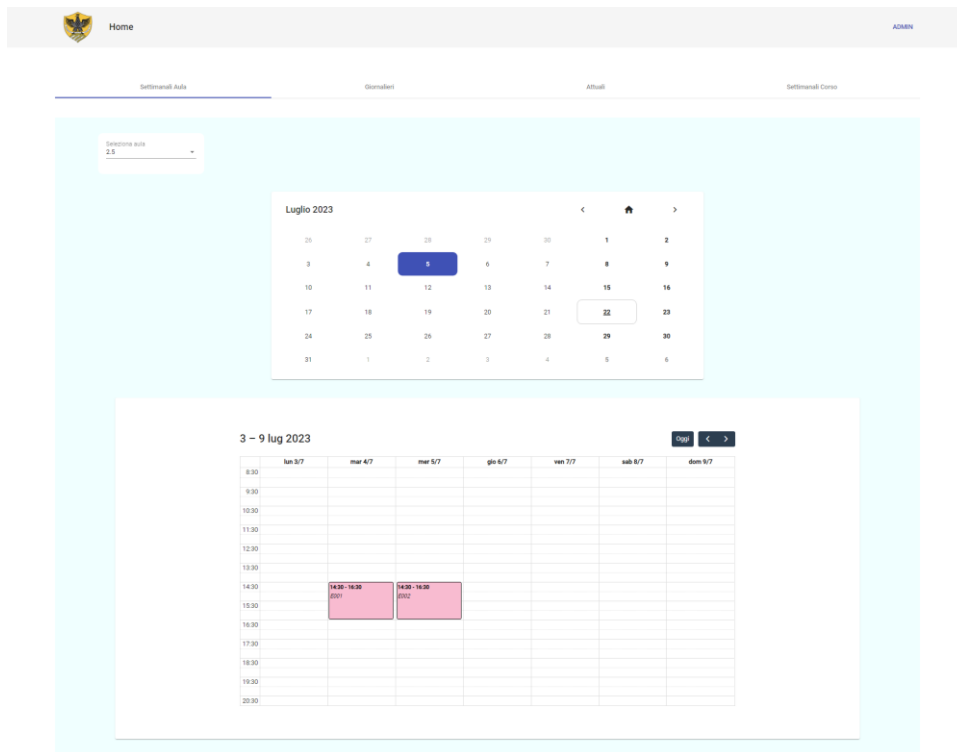
Pagina Scelta Gruppi:



In questa pagina si vedono tutti i gruppi disponibili, interagendo con uno di essi si passa alla pagina dove si potranno visualizzare le informazioni di tutte le classi tutti gli eventi a loro associati di quel determinato gruppo.

passa con

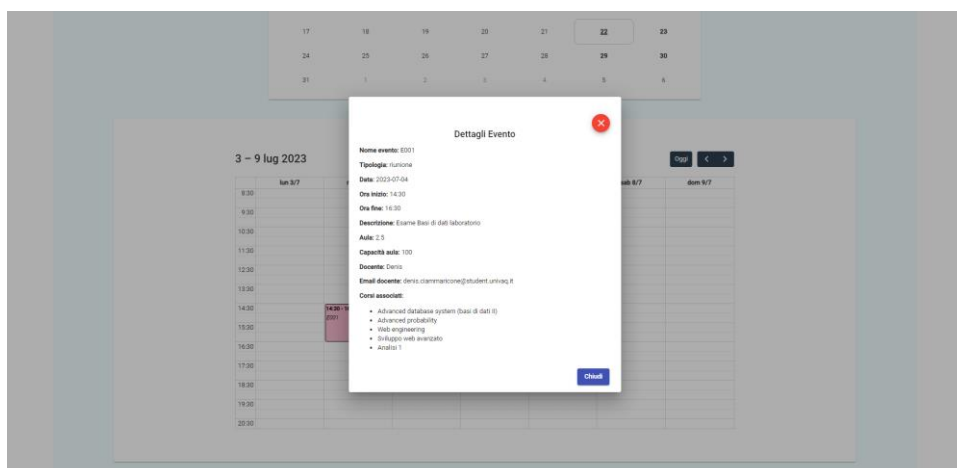
Pagina Classi ed Eventi:



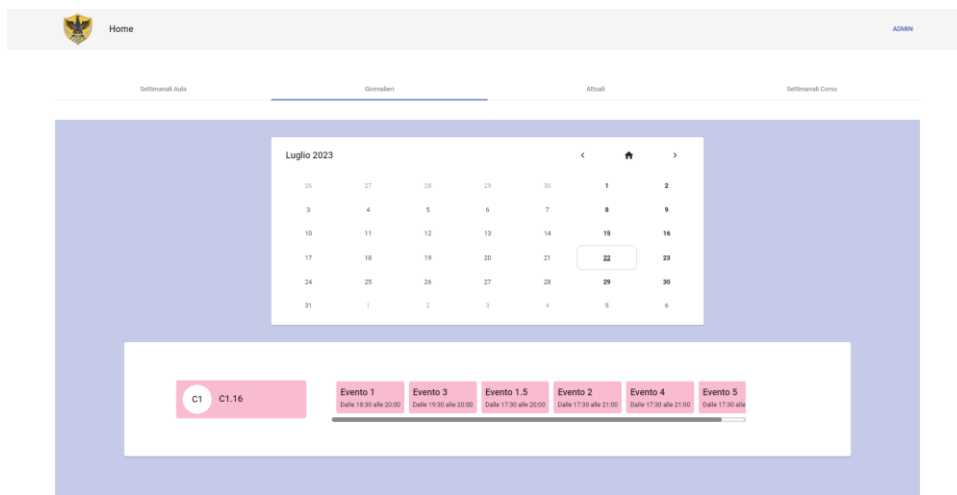
In questa pagina abbiamo 4 tasti “Settimanali Aula”, “Giornalieri”, “Attuali”, “Settimanali Corso”.

Scegliendo il primo “Settimanali Aula” otteniamo come risultato un menù a tendina in alto a sinistra “Seleziona aula” per scegliere la classe di cui ci interessa sapere gli eventi ad essa associati. Al centro dello schermo abbiamo un calendario da cui selezionare un data che, come risultato, ci genera il settimanale degli eventi associati alla nostra aula precedentemente selezionata.

Andando ad interagire con gli eventi possiamo avere maggiori informazioni su quest’ultimo.

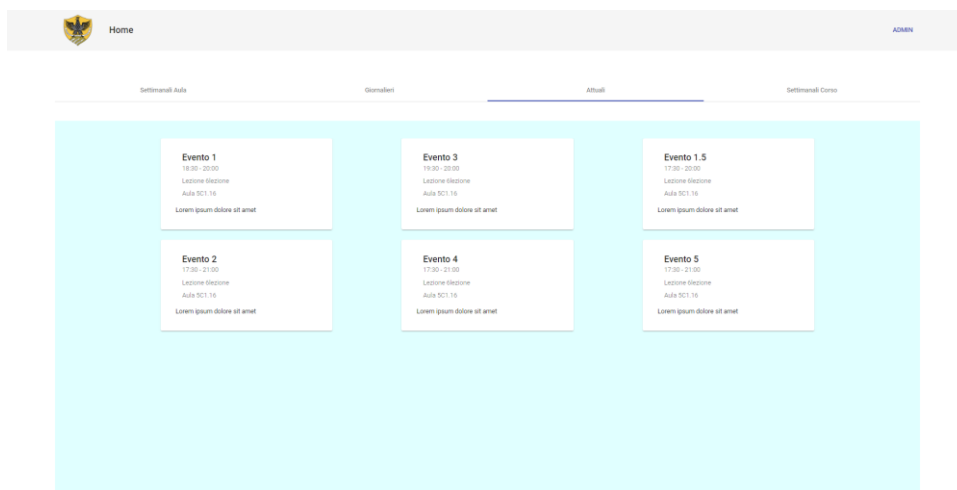


Scegliendo il secondo “Giornalieri” otteniamo



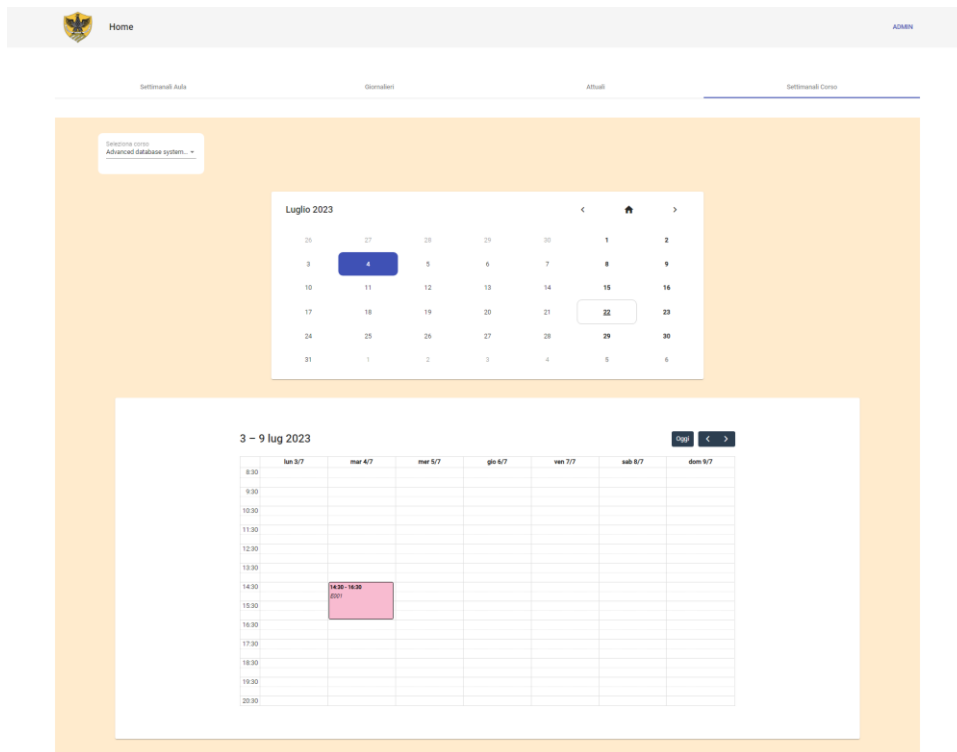
Qui otteniamo la lista degli eventi nel giorno odierno. Anche qui interagendo con gli eventi otterremo una vista con maggiori informazioni come nel caso precedente.

Scegliendo il terzo “Attuali” otteniamo



Qui otteniamo la lista degli eventi nelle prossime tre ore.

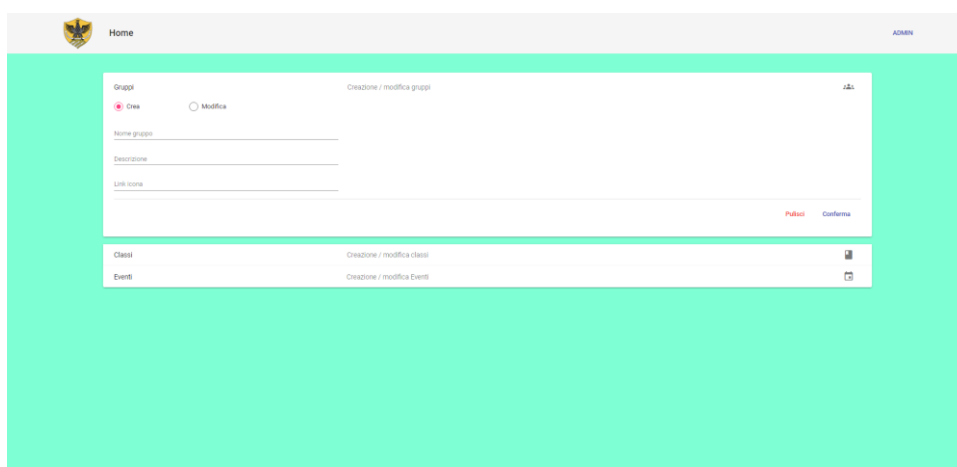
Scegliendo il quarto “Settimanali Corso” otteniamo



Qui come nel primo caso abbiamo un menù a tendina dove possiamo scegliere un Corso, il calendario dove scegliere una data. Il risultato della nostra ricerca verrà generato nel settimanale in fondo alla pagina in cui vedremo gli eventi associati al corso scelto nella data scelta.

Una volta autenticati il tasto “LOGIN” viene sostituito con il tasto “ADMIN” se vi interagiamo avremo 2 opzioni quella di logout in cui effettueremo la disconnessione e verremo riportati sulla pagina di login oppure quella di andare nella dashboard in cui poter inserire e modificare nuovi elementi all’interno del sito.

Pagina Dashboard:



In questa pagina abbiamo tre scelte “Creazione/Modifica gruppi”, “Creazione/Modifica classi”, “Creazione/Modifica eventi”.

Indipendentemente da quale sceglieremo avremo due possibilità, scegliere di creare un nuovo elemento del tipo precedentemente scelto oppure la modifica di uno già esistente.

Nel caso della creazione avremo un form da riempire con tutte le informazioni dell'entità scelta una volta riempiti tutti basterà premere il tasto "Conferma" per creare il nuovo oggetto.

Nel caso della modifica avremo un menù a tendina in cui andare a scegliere tra gli oggetti già esistenti quello che vogliamo modificare. Fatto questa anche qui bisogna compilare il form con i nuovi dati ed infine premere "Conferma" per confermare la modifica dell'oggetto.

Il tasto "Pulisci" ci permette di pulire i form in maniera rapida.

Contributo partecipanti:

Stefano si è occupato principalmente dello sviluppo del front-end e ha dato supporto per lo sviluppo del back-end.

Denis si è occupato principalmente dello sviluppo del back-end e supporto alla documentazione.

Lorenzo si è occupato principalmente dello sviluppo del database e della documentazione e ha dato un supporto allo sviluppo back-end.