Z-Planning

documentazione tecnica

Caldarone Ilaria – Cerri Sara – D'Aranno Rebecca – Faggianelli Anastasia - Ruggeri Elena

Indice

Nuxt JS/dipendenze

Nuxt 3.10

Descrizione

Nuxt.js è un framework basato su Vue.js che permette di sviluppare applicazioni web universali, ovvero sia lato client che lato server. La versione 3.10 di Nuxt introduce numerosi miglioramenti, tra cui il supporto per la Composition API di Vue 3, miglioramenti nelle performance, e una configurazione semplificata.

Link ufficiale

- Sito ufficiale di Nuxt

Installazione

npm install nuxt@3.10

V-Calendar 3.10

Descrizione

V-Calendar è una libreria di calendario per Vue.js che permette di integrare facilmente calendari interattivi nelle applicazioni. La versione 3.10 è la più recente, e include nuove funzionalità per personalizzare e integrare i calendari nelle tue pagine web.

Link ufficiale

- Sito ufficiale di V-Calendar

Installazione

V-Calendar è stato installato come dipendenza del progetto con il seguente comando:

npm install v-calendar@next

Pinia

Descrizione

Pinia è una libreria di gestione dello stato per Vue 3, sviluppata per essere utilizzata come alternativa a Vuex. È stata progettata per essere semplice, scalabile e altamente integrata con Vue 3. Pinia è particolarmente utile quando si ha bisogno di gestire lo stato globale nell'applicazione.

Link ufficiale

- sito ufficiale Pinia
- Sito ufficiale di Pinia

Installazione

npm install pinia

STORES

Store/models

Categoria

```
interface Categoria {
  id_categoria: String;
  nome: String;
  descrizione: String;
  livello: number;
}
```

Utente

```
interface Utente {
  id_utente: number;
  nome: String;
  username: String;
  cognome: String;
  genere: String;
  livello: number;
  id_coordinatore: number;
}
```

Postazione

```
interface Postazione {
  id_postazione: number;
```

```
  nome: String;
  id_categoria: String;
  descrizione: String;
  stato: number;
}
```

Prenotazione

```
interface Prenotazione {
  id_prenotazione: number;
  id_utente: number;
  id_postazione: number;
  data: Date;
  n_modifiche: number;
  flag: number;
}
```

Utente

```
interface Utente {
  id_utente: number;
  nome: String;
  username: String;
  cognome: String;
  genere: String;
  livello: number;
  id_coordinatore: number;
}
```

Store/prenotazioni.ts

Stato

prenotazioni: Prenotazione[]

---

getPrenotazioni(): Promise<void>

Recupera le prenotazioni dal server per l'utente loggato.

- Se l'utente ha livello 3 (admin), carica tutte le prenotazioni presenti sul database.
- Ordina le prenotazioni per data.
- Richiede validazione della sessione

API chiamata:

- POST getPrenotazioni.php per utenti di livello 1 e 2
- POST getPrenotazioniAdmin.php per admin (livello 3)

---

nuovaPrenotazione(data: Date, id_postazione: number): Promise<void>

Crea una nuova prenotazione per l'utente corrente.
Invia id_utente, id_postazione e data al server.

- Dopo l'inserimento, aggiorna le prenotazioni e le ordina.
- Reindirizza alla pagina "home".

Parametri:

- data: Data della prenotazione
- id_postazione: ID della postazione da prenotare

API chiamata:

- POST insertPrenotazioni.php

---

modificaPrenotazione(prenotazione: Prenotazione): Promise<void>

Modifica una prenotazione esistente.
Invia le modifiche al backend (id_prenotazione, data, id_postazione, n_modifiche).

- Dopo la modifica, ricarica le prenotazioni e reindirizza alla "home".

API chiamata:

- POST /updatePrenotazione.php

---

eliminaPrenotazione(id: number): Promise<void>

Elimina la prenotazione con l'id fornito.
Rimuove la prenotazione dallo stato locale e aggiorna i dati da server.

API chiamata:

- POST eliminaPrenotazione.php

---

filtraData(data: string): Prenotazione[]

Filtra le prenotazioni aventi una determinata data.

Parametri:

- data: Data in formato stringa

---

filtraCategoria(categoria: string): Prenotazione[]

Filtra le prenotazioni in base alla categoria della postazione.

Parametri:

- categoria: ID della categoria da filtrare

Nota: Richiede accesso allo store postazioni-store.

---

ordinaData(): void

Ordina l'array prenotazioni in ordine decrescente per data.

---

getPrenotazioneById(id: number): Prenotazione | undefined

Restituisce una copia della prenotazione corrispondente all'id fornito.

---

abilita(id_prenotazione: string): Promise<void>

Abilita una prenotazione specifica.

Effetti collaterali: Aggiorna lo stato ricaricando tutte le prenotazioni.

API chiamata:

- POST /abilitaPrenotazione.php

---

disabilita(id_prenotazione: string): Promise<void>

Disabilita una prenotazione specifica.

Effetti collaterali: Aggiorna lo stato ricaricando tutte le prenotazioni.

API chiamata:

- POST /disabilitaPrenotazione.php

---

getDatePrenotate(): string[]

Restituisce un array con le date di tutte le prenotazioni presenti nello stato.

Dipendenze esterne

- useAuth(): per recuperare utente, address, validare la sessione
- usePostazioni(): per ottenere info su postazioni e categorie
- $fetch: wrapper Nuxt per fetch API
- useRouter(): per effettuare redirect lato client

Store/postazioni.ts

Stato

postazioni: Postazione[]

categorie: Categoria[]

occupate: any[]

disabilitate: any[]

---

getPostazioni(): Promise<void>

Carica l'elenco delle postazioni dal backend (solo se non già caricato).

API chiamata:

- POST getPostazioni.php

---

getCategorie(): Promise<void>

Carica l'elenco delle categorie (se non già caricate).

API chiamata:

- POST getCategorie.php

---

checkPostazioniOccupate(data: Date): Promise<void>

Controlla quali postazioni sono occupate in una certa data.

API chiamata:

- POST checkPostazioniOccupate.php

---

getCategoria(postazione: Postazione): Categoria | undefined

Restituisce la categoria associata a una data postazione.

Parametri: Postazione

Output: Categoria associata

---

getPostazione(prenotazione: Prenotazione): Postazione | undefined

Restituisce la postazione associata a una prenotazione.

Parametri: Prenotazione

Output: Postazione associata

---

getPostazioneById(id: number): Postazione | undefined

Restituisce una postazione dato il suo ID.

Parametri: id numerico

Output: Postazione corrispondente

---

abilita(id_postazione: string): Promise<void>

Abilita una postazione (riattivandola).

API chiamata:

- POST /abilitaPostazione.php

---

disabilita(id_postazione: string): Promise<void>

Disabilita una postazione (la rende non prenotabile).

API chiamata:

- POST /disabilitaPostazione.php

---

getPostazioniDisabilitate(): Promise<void>

Recupera la lista delle postazioni disabilitate.

API chiamata:

- POST getPostazioniDisabilitate.php

---

getDateOccupate(id_postazione: number): Promise<string[]>

Restituisce un array di date in cui una postazione è già occupata.

API chiamata:

- POST getDateOccupate.php

Dipendenze esterne

- **useAuth()**: per autenticazione, sessione e indirizzo API
- **$fetch**: per effettuare le chiamate HTTP
- **Modelli:** Postazione, Categoria, Prenotazione

Store/dipendenti.ts

Stato

dipendenti: Utente[] // Lista dei dipendenti caricati

---

getDipendentiCoordinatore(id: number): Promise<void>

Recupera la lista dei dipendenti associati a un coordinatore.

API chiamata:

- POST getDipendentiCoordinatore.php

Filtra i dipendenti in base a una parola chiave (username o id_utente).

Input: stringa di ricerca

Output: Lista di Utente[] filtrati

Match case-insensitive:

- username
- id_utente

---

getPrenotazioniDipendente(id: string): Promise<Prenotazione[]>

Restituisce tutte le prenotazioni fatte da un dipendente.

API chiamata:

- POST getPrenotazioni.php

---

insertUtente(utente: Utente, password: string): Promise<any>

Inserisce un nuovo utente nel sistema.

API chiamata:

- POST insertUtente.php

---

updateUtente(utente: Utente, password: string): Promise<void>

Aggiorna le informazioni di un utente esistente.

API chiamata:

- POST updateUtente.php

Request body: identico a insertUtente

---

getDipendenteById(id: number): Utente | undefined

Restituisce un oggetto Utente cercato per id.

Input: ID utente

Output: Copia dell'oggetto Utente trovato (o undefined)

---

getCoordinatori(): Utente[]

Restituisce la lista degli utenti che hanno livello == 2 (coordinatori).

Output: Lista di Utente[]

Dipendenze esterne

- useAuth(): per gestione sessione e indirizzo API
- $fetch: per eseguire richieste HTTP
- Modelli: Prenotazione, Utente

Store/auth.ts

gestisce autenticazione, sessione utente, login/logout, e persistenza locale (localStorage).

Stato

utente: Utente // Utente loggato

sessione: {

id_sessione: number, // 0 = non attiva

scadenza: number,

controlCode: number // Token anti-bruteforce

}

---

address: string // URL base per le API

init(): void

Inizializza i dati dallo localStorage (se presenti).
Serve a ripristinare la sessione utente dopo un refresh.

Effetti: Popola utente e sessione

---

login(id: number, password: string): Promise<string | void>

Effettua il login dell'utente.

API chiamata:

- POST login.php

controllaSessione(): Promise<boolean>

Verifica se la sessione è ancora valida.

API chiamata:

- POST sessionCheck.php

---

logout(): Promise<boolean>

Chiude la sessione dell'utente.

API chiamata:

- POST logout.php

---

setLocalStorage(): void

Salva utente e sessione su localStorage.

Effetti: Persiste i dati della sessione tra i refresh

---

clearLocalStorage(): void

Rimuove utente e sessione dal localStorage.

---

testaPassword(password: string): boolean

Verifica la robustezza della password secondo 5 criteri:

- Almeno 8 caratteri
- Almeno una **maiuscola**
- Almeno una **minuscola**
- Almeno un **numero**
- Almeno un **carattere speciale**

Output: true se valida, altrimenti false

Dipendenze esterne

- useRouter() per il redirect
- $fetch per le chiamate HTTP
- localStorage per la persistenza client-side
- Utente model

API

API/abilitaPostazione.php

Questa API aggiorna lo stato di una **postazione** a "abilitata" (stato = 0) e azzera i flag delle **prenotazioni** segnalate come disabilitate a causa di postazioni disabilitate (flag = 2);

Body della richiesta (JSON)

- Id_postazione

API/abilitaPrenotazione.php

Questa API consente di **attivare** una prenotazione specifica impostando il campo flag a 0.

Body della richiesta (JSON)

- Id_prenotazione

API/checkPostazioniOccupate.php

estituisce tutte le postazioni occupate per una determinata data. Body della richiesta (JSON)

Body della richiesta (JSON)

- Data

API/disabilitaPostazione.php

aggiorna lo stato di una **postazione** a "disabilitata" (stato = 1) e imposta i flag delle **prenotazioni** segnalate coinvolte(flag = 2);

Body della richiesta (JSON)

- Id_postazione

API/abilitaPrenotazione.php

Questa API consente di **disattivare** una prenotazione specifica impostando il campo flag a 1.

Body della richiesta (JSON)

- Id_prenotazione

API/eliminaPrenotazione.php

Questa API consente di eliminare una prenotazione specifica.

Body della richiesta (JSON)

- Id_prenotazione

API/getCategorie.php

restituisce un elenco di tutte le **categorie** presenti nel sistema.

API/getDateOccupate.php

Restituisce **tutte le date** delle prenotazioni effettuate per una specifica **postazione**, identificata dal suo id_postazione.

Body della richiesta (JSON)

- Id_postazione

API/getDipendentiCoordinatore.php

Recupera l'elenco degli **utenti** associati a un **coordinatore** specifico (id_coordinatore). Se il coordinatore ha livello 3 (amministratore), restituisce **tutti** gli utenti.

Body della richiesta (JSON)

- Id_utente

API/getPostazioni.php

restituisce un elenco di tutte le **postazioni** presenti nel sistema.

API/getPostazioniDisabilitate.php

restituisce un elenco di tutte le **postazioni disabilitate** (flag = 1) presenti nel sistema.

API/getPrenotazioni.php

Restituisce tutte le **prenotazioni effettuate** da un utente specifico (id_utente), ordinate dalla più recente alla meno recente.

Body della richiesta (JSON)

- Id_utente

API/getPrenotazioniAdmin.php

restituisce un elenco di tutte le **prenotazioni** presenti nel sistema (riservato agli admin).

API/getUtenteById.php

Recupera le informazioni personali di un utente identificato dal proprio id_utente.

Body della richiesta (JSON)

- Id_utente

API/insertPrenotazioni.php

Inserisce una nuova prenotazione per una determinata postazione, da parte di un utente, in una data specifica.

Body della richiesta (JSON)

- Id_utente
- Id_postazione
- data

API/insertUtente.php

Inserisce una nuova prenotazione per una determinata postazione, da parte di un utente, in una data specifica.

Body della richiesta (JSON)

- nome
- cognome
- username
- genere
- password
- livello
- id_coordinatore

API/login.php

Questa API consente a un utente di autenticarsi nel sistema. Viene effettuato il controllo della password e, se corretta, viene generata una sessione con un identificativo unico e un codice di controllo.

Body della richiesta (JSON)

- Id_utente
- password
- id_sessione

API/logout.php

Questa API consente a un utente di effettuare il logout dal sistema. La sessione viene validata utilizzando il controlCode, e se il codice di controllo è corretto, la sessione viene distrutta.

Body della richiesta (JSON)

- controlCode
- id_sessione

API/sessionCheck.php

Questa API verifica se la sessione dell'utente è ancora valida. Se la sessione è scaduta o il controlCode non è corretto, la sessione viene distrutta. Se la sessione è valida, viene restituito un messaggio di conferma.

Body della richiesta (JSON)

- controlCode
- id_sessione

API/updatePasswrd.php

Questa API consente a un utente di aggiornare la propria password nel sistema. Il client invia l'ID dell'utente e la nuova password, e l'API aggiorna il campo della password nel database.

Body della richiesta (JSON)

- id_utente
- password

API/updatePrenotazione.php

Questa API consente di aggiornare i dettagli di una prenotazione già esistente, inclusi l'ID della postazione, la data e il numero di modifiche effettuate. La richiesta richiede l'ID della prenotazione,

la nuova postazione, la nuova data e il numero di modifiche, che viene incrementato automaticamente.

Body della richiesta (JSON)

- id_prenotazione
- id_postazione
- data
- n_modifiche

API/updateUtente.php

Questa API consente di aggiornare i dettagli di un utente esistente nel database. I dati da aggiornare includono il nome, il cognome, il genere, il username, la password, il livello dell'utente e l'ID del coordinatore.

Body della richiesta (JSON)

- nome
- cognome
- username
- genere
- password
- livello
- id_coordinatore
