Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Analisi, progettazione e sviluppo del backend di un'applicazione web per la gestione di eventi

Tesi di laurea

Relatore
Prof.Davide Bresolin

Laureando
Alberto Lazari

Anno Accademico 2021-2022

Alberto Lazari: *Analisi, progettazione e sviluppo del backend di un'applicazione web per la gestione di eventi,* Tesi di laurea, © Luglio 2022.

Sommario

La tesi descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, presso la sede di Treviso di Moku S.r.l., il cui obiettivo era la reimplementazione del backend di una piattaforma di gestione di eventi, sfruttando gli strumenti tipicamente utilizzati nei progetti dell'azienda.

In particolare, i seguenti capitoli tratteranno del contesto lavorativo dell'azienda, dell'analisi svolta sullo stato della piattaforma ad inizio stage, della progettazione e successiva implementazione iniziale del nuovo backend, focalizzando l'attenzione sulle scelte stilistiche e architetturali perseguite.

Ringraziamenti

Voglio ringraziare il Prof. Davide Bresolin, per l'interesse, il supporto e l'aiuto fornito durante il periodo di stage e di stesura di questa tesi.

Ringrazio i miei genitori e tutti i miei famigliari per il supporto e l'affetto che mi hanno donato durante questi hanno di studio.

Ringrazio i colleghi di Moku che mi hanno accolto calorosamente tra loro durante la mia esperienza di stage, in particolare Riccardo e Nicolò, per avermi sempre fornito l'aiuto che cercavo durante il mio lavoro.

Ringrazio la Comunità Capi del Mestre 2, per avermi accompagnato fin qui, infondendo in me una maturità e una consapevolezza che mi hanno permesso di raggiungere questo traguardo.

Ringrazio i miei colleghi studenti, stagisti e gli amici dei gruppi di progetto, con cui ho condiviso le difficoltà e le soddisfazioni di quest'anno.

Infine ringrazio i miei amici, che mi sono sempre stati vicini e con cui ho condiviso esperienze indimenticabili.

Padova, Luglio 2022 Alberto Lazari

Indice

Elenco delle figure

Elenco delle tabelle

Elenco degli esempi di codice

Capitolo 1

L'azienda

1.1 Descrizione generale

Descrizione dell'azienda: brevissima storia, divisione dei ruoli, spazi e luoghi di lavoro.

1.2 Modello di sviluppo

Modello agile, Scrum, organizzazione dei team.

Capitolo 2

Descrizione dello stage

2.1 Introduzione al progetto

Storia del progetto prima del mio arrivo, azienda che ha commissionato il progetto, descrizione dello scopo della piattaforma e del suo funzionamento, motivazioni alla base della scelta di riscrittura del backend.

2.2 Requisiti

Requisiti obbligatori, desiderabili e opzionali previsti.

2.3 Pianificazione

Divisione settimanale del lavoro dal piano di lavoro, incluse correzioni.

2.4 Tecnologie utilizzate

Descrizione della configurazione del framework Ruby on Rails utilizzata: librerie utilizzate, Postgres, AWS, API REST .

Capitolo 3

Analisi e refactor dei modelli

3.1 Introduzione

Spiegazione del lavoro svolto in questa fase.

3.2 Modifiche effettuate

Decisioni significative prese durante l'attività di refactor dei modelli.

3.3 Diagramma ER completo

Diagramma ER del nuovo backend.

Capitolo 4

Progettazione della API

4.1 Introduzione

L'API è la parte più importante del backend: è l'interfaccia che viene esposta per interagire con l'applicazione. Viene utilizzata dal frontend per ottenere i dati necessari a soddisfare le richieste degli utenti, che, essendo già sviluppato, si aspetta di poter eseguire certe richieste e di ricevere una risposta di un certo tipo. È previsto che il frontend debba adattarsi ai cambiamenti del nuovo backend, una volta pronto, ma questi dovrebbero essere ridotti al minimo, quindi nella progettazione dei controller ho cercato di trovare il giusto equilibrio con quanto era stato fatto in precedenza e lo stile che viene suggerito dal framework.

4.2 Notazione adottata

Prima di descrivere la struttura delle operazioni viene illustrata la notazione adottata per gli endpoint:

- la prima parte indica il metodo HTTP utilizzato: uno tra GET, POST, PUT, PATCH e DELETE, negli endpoint realizzati;
- la seconda parte indica l'URI della risorsa relativa all'endpoint, senza specificare l'URL dell'applicazione;
- tutti gli URI degli endpoint implementati sono all'interno del percorso `/api`;
- alcuni endpoint si riferiscono a una specifica risorsa invece che a una lista o collezione. In tal caso l'identificativo della risorsa viene rappresentato con la notazione `:id` o `:resource_id`;
- viene utilizzata la parola `resources` per rappresentare il nome di un modello generico.

Ad esempio:

1. una chiamata con metodo HTTP PUT a `https://app.evvvents.it/api/platforms/2` risponde all'endpoint `PUT /api/platforms/:id`;
2. a un endpoint generico `DELETE /api/resources/:id` può rispondere alle chiamate `DELETE /api/locations/1` o `DELETE /api/users/4`.

Ad ogni risposta viene associato uno stato HTTP. In caso di successo viene sempre restituito il codice `200`, mentre gli errori restituiscono codici diversi, in base al contesto. I codici di errore genrali, che possono essere restituiti da qualsiasi endpoint implementato sono:

- `404`, nel caso in cui l'endpoint o la risorsa richiesta non esista;
- `500`, nel caso di errori generici nel backend, che non sono stati gestiti esplicitamente.

Gli altri casi di errore vengono descritti nelle sezioni successive.

4.3 Descrizione delle funzionalità esposte

Di seguito vengono descritte le operazioni che deve essere possibile eseguire su ogni modello dell'applicazione, chiamando i relativi endpoint.

4.3.1 Lista delle risorse

Risponde all'endpoint `GET /api/resources`, restituendo la lista delle risorse associate al modello specificato, generalmente ordinate per nome, dove presente. La risposta prevede l'utilizzo della paginazione, con cui si può interagire attraverso i due parametri GET opzionali:

1. `page`, che indica il numero della pagina che si vuole visualizzare. Se non indicato, la risposta mostrerà la prima pagina;
2. `per_page`, che indica il numero di risorse presenti per pagina. Se non indicato vengono mostrate 25 risorse per pagina, valore già utilizzato nel frontend.

Di ogni risorsa vengono mostrate solo le informazioni essenziali, invece di tutte quelle fornite dal modello. Di seguito vengono elencati gli attributi restituiti per ogni risorsa, nella lista di ogni modello implementato:

User

- `id`,
- `email`,
- `first_name`,
- `last_name`,
- `role`,
- `organizer_id`

Plan

- `id`,
- `name`,
- `description`,
- `price`

Platform

- `id`,
- `name`,
- `website`,
- `host_url`,
- `main_organizer_id`

Organizer

- `id`,
- `name`,
- `platform_id`,
- `address`,
- `city`,
- `country`

Location

- `id`,
- `name`,
- `organizer_id`

4.3.2 Dettagli di una risorsa

Risponde all'endpoint `GET /api/resources/:id`, restituendo i dettagli della risorsa specificata. I dettagli non comprendono la totalità degli attributi del modello, ma un suo sottinsieme selezionato, per nascondere all'utente eventuali dettagli implementativi non rilevanti o attributi di tracciamento, come il creatore o l'istante di creazione e ultima modifica del record. Di seguito vengono elencati, per ogni modello, gli attributi restituiti per ogni risorsa:

User

- `id`,
- `email`,
- `image`,
- `first_name`,
- `last_name`,
- `role`,
- `phone`,

- `status`,
- `organizer_id`

Plan

- `id`,
- `name`,
- `price`,
- `description`,
- `max_events`,
- `max_participants`,
- `max_integrations`,
- `max_storage`,
- `desk_number`

Platform

- `id`,
- `name`,
- `logo`,
- `logo_dark`,
- `favicon`,
- `logo_small`,
- `website`,
- `host_url`,
- `privacy_url`,
- `powered_by`,
- `action_bar_color`,
- `side_bar_color`,
- `border_color`,
- `sign_in_color`,
- `event_color`,
- `status`,
- `plan_id`,
- `main_organizer_id`

Organizer

- `id`,
- `name`,
- `logo`,
- `platform_id`,
- `vat`,
- `sdi`,
- `address`,
- `city`,
- `country`,
- `zip_code`,
- `status`,
- `email`,
- `email_pec`,
- `phone`,
- `finished_webhook`,
- `add_participant_webhook`,
- `remove_participant_webhook`

Location

- `id`,
- `name`,
- `status`,
- `description`,
- `image`,
- `address`,
- `civic`,
- `city`,
- `province`,
- `zip_code`,
- `google_place_identifier`,

- `latitude`,
- `longitude`,
- `how_to_get_by_car`,
- `how_to_get_by_plane`,
- `how_to_get_by_train`,
- `internal_notes`,
- `organizer_id`

4.3.3 Creazione di una risorsa

Risponde all'endpoint `POST /api/resources`, creando un nuovo record del relativo modello, con gli attributi specificati nel corpo della richiesta. Restituisce nella risposta i dettagli della risorsa appena creata, se l'operazione è avvenuta con successo.

4.3.4 Modifica di una risorsa

Risponde agli endpoint `PUT /api/resources/:id` e `PATCH /api/resources/:id`, modificando gli attributi della risorsa, specificati nel corpo della richiesta, con i valori passati. Restituisce nella risposta i dettagli della risorsa appena modificata, se l'operazione è avvenuta con successo.

4.3.5 Eliminazione di una risorsa

Risponde all'endpoint `DELETE /api/resources/:id`, eliminando la risorsa fisicamente dal database oppure logicamente, impostando lo stato del record ad `archived`, nei modelli in cui è prevista questa modalità di cancellazione. Restituisce nella risposta i dettagli della risorsa appena eliminata, se l'operazione è avvenuta con successo.

4.3.6 Lista dei ruoli degli utenti

Risponde all'endpoint `GET /api/users/roles`, elencando tutti i possibili ruoli di un utente:

- super admin;
- admin;
- manager;
- staff;
- speaker;
- participant.

Questa è un'informazione utile al frontend, che utilizza già estensivamente, quindi andava fornita attraverso un endpoint dedicato, non potendo essere reperita in altri modi. Per la struttura attuale dell'applicazione la risposta è sempre la stessa, perché i ruoli sono *hard coded* nel software.

4.3.7 Lista delle risorse interne a una specifica risorsa

Alcuni modelli sono logicamente contenuti all'interno di altri, si vuole quindi poter elencare tutte le risorse contenute in una seconda risorsa. Nei modelli che sono stati implementati durante il progetto di stage questo accade in due casi:

1. gli organizzatori stanno all'interno di una specifica piattaforma;
2. ogni organizzatore crea e utilizza delle *location*.

In questi casi se si vogliono elencare le risorse `rescources_b` contenute nella risorsa `resource_a` con identificativo `:id` si effettua una chiamata all'endpoint `GET /api/resources_a/:id/resources_b`, ad esempio per elencare tutti gli organizzatori della piattaforma con id 10 si chiama `GET /api/platforms/10/organizers`. È comunque possibile ottenere la lista di tutte le risorse relative a un modello, anche quando questo stia logicamente all'interno di un'altra risorsa. Ad esempio è comunque possibile ottenere la lista di tutti gli organizzatori di tutte le piattaforme chiamando `GET /api/organizers`.

4.3.8 Creazione di una risorsa all'interno di un'altra risorsa

Nei casi appena sopracitati è necessario specificare la risorsa in cui sarà contenuta quella che si intende creare, quindi ho ritenuto oppurtuno rendere necessaria la creazione di queste particolari risorse chiamando l'endpoint `POST /api/resources_a/:id/resources_b`, invece che con l'endpoint descritto nella sotto-sezione 4.3.3, che non viene esposto per i modelli interessati.

4.4 Gestione dei permessi

L'API, oltre ad essere utilizzata dal frontend, potrebbe essere usata da qualsiasi altro tipo di *client*, con o senza buone intenzioni, quindi è fondamentale che siano presenti dei controlli sulle autorizzazioni degli utenti che effettuano richieste, per ogni funzionalità esposta.

4.4.1 Autenticazione e autorizzazione

Prima di procedere bisogna fare un'importante premessa: autenticazione e autorizzazione, seppur simili, sono due concetti diversi e, come tali, devono essere separati anche strutturalmente.

L'autenticazione avviene quando l'utente esegue l'azione tecnicamente chiamata *sign-in* o *log-in*. Serve a provare che l'utente è chi dichiara di essere. A livello di codifica questa viene gestita da Devise, una gemma adatta allo scopo, dunque non è stato necessario definirne la progettazione.

L'autorizzazione definisce se uno specifico utente o tipologia di utenti abbia o meno il permesso di eseguire una certa azione. Questa sezione si occupa di descrivere le autorizzazioni richieste per ogni funzionalità implementata dall'API, riferita nel modo seguente:

- **lista**: si riferisce a quanto descritto in §4.3.1;
- **lista nella risorsa**: si riferisce a quanto descritto in §4.3.7;
- **dettaglio**: si riferisce a quanto descritto in §4.3.2;

- **creazione**: si riferisce a quanto descritto in §4.3.3 e in §4.3.8;
- **modifica**: si riferisce a quanto descritto in §4.3.4;
- **eliminazione**: si riferisce a quanto descritto in §4.3.5.

Nella maggior parte dei casi non sarà necessario entrare così nel dettaglio, perché spesso le azioni di lista e di dettaglio sono accomunate nella categoria delle azioni di lettura, mentre quelle di creazione, modifica ed eliminazione vengono categorizzate come azioni di scrittura, tutte con gli stessi permessi di base, a cui eventualmente si aggiungono ulteriori vincoli o permessi.

4.4.2 Permessi richiesti

I permessi richiesti dalle azioni sui modelli vengono definite in base al ruolo dell'utente autenticato nel sistema al momento della richiesta. La prima condizione per qualsiasi azione quindi, è che l'utente abbia eseguito con successo l'autenticazione (azione per cui ovviamente la condizione non vale). L'azione stessa di autenticazione richiede delle autorizzazioni: sono tutte gestite da Devise, tranne la condizione che prevede che un utente non possa eseguire il *log-in* se ha il ruolo di partecipante, perché l'applicazione web è dedicata solamente alla gestione degli eventi, a cui i partecipanti non devono avere accesso: loro si interfacciano al sistema esclusivamente attraverso l'applicazione *mobile*.

User

L'autorizzazione delle azioni eseguite da un utente su un altro utente sono particolarmente complicate, perché dipendono dalla combinazione dei ruoli dell'utente interessato e dell'utente che ha eseguito la richiesta.

- **lettura**: tutti gli utenti possono eseguire questa azione, ognuno però vede solamente gli utenti diversi da se stesso, con ruolo uguale o inferiore all'utente che ha eseguito la richiesta. A questo si aggiunge che:
  - l'admin vede solamente utenti della sua stessa piattaforma;
  - il manager e lo staff vedono solamente utenti della loro stessa organizzatore;
  - lo speaker vede solo partecipanti e non può essere visto da nessuno, perché dipende dal modello degli eventi, che non è stato implementato durante lo svolgimento del progetto di stage.
- **dettaglio**: ai permessi di lettura si aggiunge il permesso di un utente di vedere i dettagli di se stesso;
- **scrittura**: i permessi sono gli stessi della lettura, ma in questo caso il ruolo dell'utente che esegue la richiesta deve essere superiore in senso stretto;
- **modifica**: ai permessi di scrittura si aggiunge il permesso, per un utente, di modificare se stesso;
- **eliminazione**: ai permessi di scrittura si aggiunge il permesso, per un utente, di eliminare se stesso, nel solo caso in cui questo sia uno speaker;
- **lista dei ruoli**: si riferisce a §4.3.6. Non è richiesta né l'autorizzazione, né l'autenticazione.

Plan

- **lettura**: concessa a tutti gli utenti;
- **scrittura**: permessa solamente ai super admin.

Platform

- **lista**: permessa solamente ai super admin;
- **dettaglio**: permesso ai super admin e agli admin della piattaforma su cui viene eseguita la richiesta;
- **scrittura**: permessa solamente ai super admin.

Organizer

- **lista**: permessa solamente ai super admin;
- **lista nella piattaforma**: permessa ai super admin e agli admin della piattaforma;
- **dettaglio**: come la lista nella piattaforma, ma è permesso anche al manager dell'organizzatore;
- **scrittura**: come la lista nella piattaforma.

Location

- **lista**: permessa solamente ai super admin;
- **lista nell'organizzatore**: come il dettaglio per il modello **Organizer**;
- **dettaglio**: come la lista nell'organizzatore;
- **scrittura**: come la lista nell'organizzatore;

4.4.3 Parametri permessi

L'autorizzazione deve anche occuparsi dei parametri che possono essere permessi nel corpo delle richieste di creazione e modifica delle risorse. Vengono elencati di seguito per ogni modello:

User

Per la creazione:

- `email`,
- `first_name`,
- `last_name`,
- `role`,
- `phone`,
- `password`,

- `organizer_id`,
- `attachments`:
  - `image`

Per la modifica sono permessi gli stessi parametri consentiti per la creazione, eccetto la password e, se un utente sta modificando se stesso, il ruolo.

Plan

- `name`,
- `description`,
- `max_events`,
- `max_participants`,
- `max_integrations`,
- `max_storage`,
- `desk_number`,
- `price`

Platform

- `name`,
- `website`,
- `host_url`,
- `privacy_url`,
- `powered_by`,
- `action_bar_color`,
- `side_bar_color`,
- `border_color`,
- `sign_in_color`,
- `event_color`,
- `main_organizer`,
- `plan_id`,
- `attachments`:
  - `logo`,
  - `logo_dark`,
  - `favicon`,
  - `logo_small`

Organizer

- name,
- vat,
- sdi,
- address,
- city,
- country,
- zip_code,
- email,
- email_pec,
- phone,
- finished_webhook,
- add_participant_webhook,
- remove_participant_webhook,
- attachments:
  - logo

Location

- name,
- description,
- address,
- civic,
- city,
- province,
- zip_code,
- google_place_identifier,
- latitude,
- longitude,
- how_to_get_by_car,
- how_to_get_by_plane,
- how_to_get_by_train,
- internal_notes,
- attachments:
  - image

Capitolo 5

Codifica

5.1 Modelli

5.1.1 Migrazioni del database

Comando `rails generate` *e migrazione prodotta.*

5.1.2 Associazioni a modelli e file

Associazioni di Active Record e Active Storage.

5.1.3 Validazioni

Validazioni sugli attributi del modello e le associazioni.

5.2 Controller

5.2.1 APIController

Descrizione dei metodi di utilità ereditati dai controller dell'API.

5.2.2 Implementazione delle action

Descrizione ed esempio di action tipiche dei controller.

5.3 Gestione dei permessi

Funzionamento e uso della gemma "Pundit" per la gestione dei permessi relativi agli endpoint dell'API, scope e metodi relativi alle action, esempio di gestione della gerarchia che andrà rivisto

5.4 Test di unità

Descrizione della gemma "RSpec", che fornisce strumenti per lo sviluppo guidato dal comportamento (behaviour-driven development), esempi di modelli testati.

Capitolo 6

Conclusioni

6.1 Raggiungimento dei requisiti

Tabella con stato di completamento dei requisiti, con commento (dove necessario)

6.2 Valutazione personale

Messe alla prova le competenze fornite dal corso di laurea, verificata l'efficacia dei corsi e dei progetti svolti, imparato un nuovo linguaggio e framework con filosofia di sviluppo a me nuova, scoperto ambiente lavorativo aziendale con i ruoli e le dinamiche interne.

Bibliografia