Progetto Base Antartide

Stage EURIS 2023 - Team Charmender

Abstract

La Stazione Concordia, centro di ricerca internazionale situato in antartide, ha richiesto una applicazione per gestire lo scambio di informazioni tra la comunità scientifica internazionale (csi) e il gruppo di ricerca presso la base scientifica, l'applicazione permetterà di:

- dare visibilità circa l'avanzamento degli esperimenti a livello macroscopico
- gestire a livello puntuale ogni songolo esperimento, con la possiblità di lasciare commenti e di definirne le caratteristiche

L'applicazione sarà fruibile dal personale della base scientifica con dei tablet, strumento con il quale possono sia consultare che fare modifiche agli esperimenti.

L'applicazione dovrà altresì gestire la conciliazione dei dati tra le richieste sugli esperimenti che arrivano dalla csi e l'aggiornamento degli stessi da parte dei ricercatori della stazione perché la connessione internet (fornita tramite satellite) è disponibilie per un lasso di tempo breve durante il giorno.

Requisiti

1. Allineare la base scientifica alle richieste

Quando la connessione internet è disponibile bisogna riversare i dati relativi alle richieste sui sistemi informativi della Concordia. Le richieste esterne vengono veicolate alla base tramite API Trello.

Trello è una web app che permette di organizzare il maniera schematica e dinamica il flusso di lavoro tra più persone ed ha a disposizione vari strumenti e modelli per organizzare le attività, di seguito i soli *dati* di interesse, con relativi *attributi*, che si vogliono tener in considerazione:

1. **Lista (stato avanzamento)**
    - *Titolo*
2. **Scheda (esperimento)**
    - *Titolo*
    - *Descrizione*
    - *Scadenza (se presente)*
    - *Priorità*
    - *Ultimo commento*
    - *Assegnatario dell'esperimento (se presente)*

2. Allineare la base scientifica alle richieste

Gli scienziati devono poter modificare gli esperimenti (scheda) assegnati in ogni istante del giorno, indipendentemente dalla disponibilità della connessione ad internet. Di seguito le uniche azioni eseguibili dagli scienziati sulle singole schede:

- aggiungere un commento ad un esperimento esistente (se assegnatario)
- cambiare lo stato di avanzamento (lista) dell'esperimento (se assegnatario o assegnatario assente)

3. Allineare centri di ricerca

La csi deve ricevere tramite Trello API gli aggiornamenti sulle schede degli esperimenti modificate dagli scienziati. Di seguito le uniche azioni eseguibili dai membri della csi sulle schede:

- aggiungere un commento ad un esperimento esistente
- creare un esperimento e definirne gli attributi (aggiungere l'esperimento alla lista 'Da Fare' solo dopo aver definito tutti gli attributi in una lista di transizione dedicata alla creazione di esperimenti)

4. Dispositivi in dotazione

Gli scienziati hanno a disposizione dei tablet con i quali sono connessi alla rete locale della base e con i quali, oltre tener traccia degli avanzamenti dei task assegnati, hanno in evidenza i task non conclusi con questo ordine:

1. esperimenti ad alta priorità
2. esperimenti con scadenza < 5 giorni
3. esperimenti a media priorità
4. esperimenti a bassa priorità

5. Requisiti non funzionali

1. Parametrizzare i valori con i quali avviene l'allineamneto poichè l'orario e la finestra di tempo di passaggio del satellite, e quindi della connessione con il resto del mondo, varia in base al periodo
2. Definire già i valori al punto 1 per poter coprire tutto l'anno solare senza dover metter mano al codice una volta deployato sui sistemi della concordia

Contratti REST

Endpoints

Consumando l'API Trello accederemo a questi endpoint. I token devono essere ottenuti per ogni scienziato in modo che possa essere associata al suo profilo l'azione di aggiunta commento. (Notare che tutti i Get Endpoints possono anche rispondere con oggetti nulli o vuoti)

Variabili
BaseURL = `https://api.trello.com`
ListId* = `64760975fbea80d6ef329080`
CardId* = `64761756c338bac930497a59`
Key* = `ATTAd93cf67ec0072d821ff32e199156a675ed9301feea0f899df160829b3f14082dAB1E41AD`
Token* = `9ba27d32be683843dd1ffb346ae07641`

**i valori riportati sono a titolo esemplificativo*

Get Cards On A List

Endpoint
`GET /1/lists/{id}/cards` *(l'id è l'id della lista, riportato in appsetting)*

Url Esempio
`{{BaseURL}}/1/lists/{{ListId}}/cards?key={{Key}}&token={{Token}}`

Restituisce tutti gli oggetti card su la lista con l'id specificato.
Contiene:

- members' ids
- nome card
- description
- labels
- due date

Update A Card (Moving it from list to list)
Endpoint
`PUT /1/cards/{id}` *(l'id è l'id della carta, che può essre recuperato)*

Url Esempio
`{{BaseURL}}/1/cards/{{CardId}}?idList={{ListId}}&key={{Key}}&token={{Token}}`

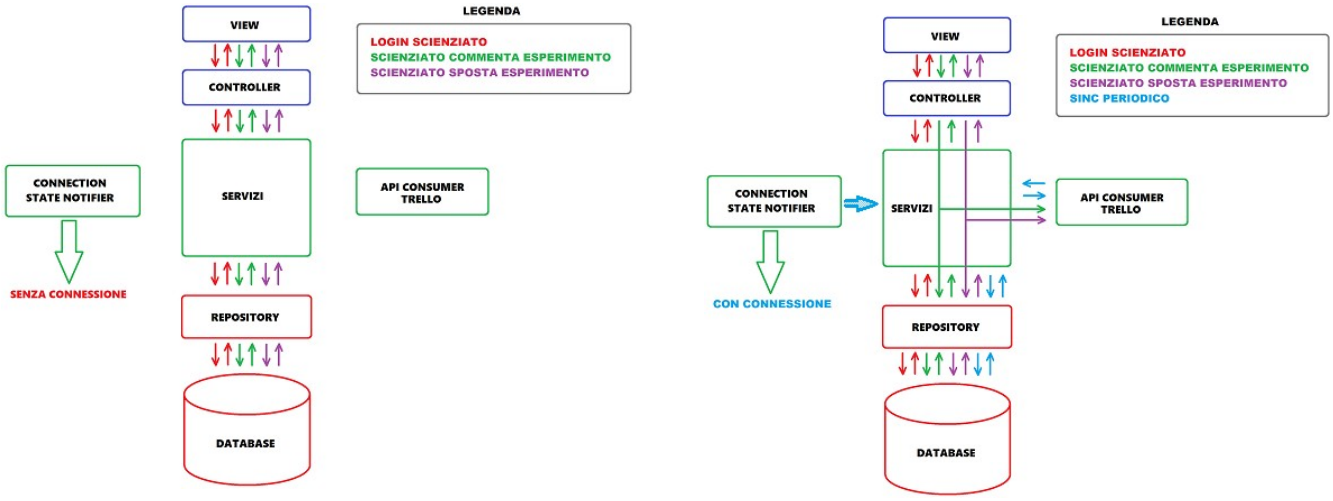
Add A Comment To A Card
Endpoint
`POST /1/cards/{id}/actions/comments`

Url Esempio
`{{BaseURL}}/1/cards/{{CardId}}/actions/comments?text=example&key={{Key}}&token={{Token}}`

Get All Comments on a Board (Could be null/empty)
Endpoint
`GET /1/boards/{id}/actions?filter=commentCard`

Url Esempio
`{{BaseURL}}/1/boards/{{BoardId}}/actions?filter=commentCard&key={{Key}}&token={{Token}}`

Soluzione Proposta



L'idea di base è che gli scienziati della base Concordia possono commentare gli esperimenti e modificarne lo stato di avanzamento senza curarsi della presenza di connessione perché le loro azioni vengono registrate nel database locale. Quando la connessione è disponibile, avviene la sincronizzazione con il database remoto, e, fintanto che la connessione rimane disponibile, le modifiche effettualte dagli scienziati vengono simultaneamente registrate sia su database locale che su database remoto chiamando opportunamente l'API Trello.

Tecnologie

Per realizzare il software che verrà dato a disposizone della Stazione Concordia il team di sviluppo ha individuato le seguenti tecnologie necessarie per la sua implementazione:

1. Trello API
2. SQL server
3. Polly - 7.2.4
    - Microsoft.Extensions.Http.Polly - 7.0.7
    - Polly.Contrib.WaitAndRetry - 1.1.1
    - Polly.Extensions.Http - 3.0.0
4. Entity Framework
    - Microsoft.EntityFrameworkCore - 7.0.5
    - Microsoft.EntityFrameworkCore.SqlServer - 7.0.5
    - Microsoft.EntityFrameworkCore.Tools - 7.0.5
    - Microsoft.EntityFrameworkCore.Design - 7.0.5
5. .NET 6.0
6. Microsoft.Extensions.Configuration.Abstractions - 7.0.0
7. Microsoft.Extensions.Hosting.Abstractions - 7.0.0
8. Microsoft.Extensions.Logging.Abstractions - 7.0.0
9. Microsoft.Extensions.Http - 7.0.0
10. Newtonsoft.Json - 13.0.1
11. Coverlet.collector - 3.2.0
12. Automapper - 12.0.1
    - AutoMapper.Extensions.Microsoft.DependencyInjection - 12.0.1
    - AutoMapper.Collection - 9.0.0
    - AutoMapper.EF6 - 2.1.1
13. xunit - 2.4.2
    - xunit.runner.visualstudio - 2.4.5
14. FluentAssertions - 6.11.0
15. System.Net.Http - 4.3.4
16. System.Net.Http.Json - 7.0.1
17. Microsoft.NET.Test.Sdk - 17.5.0
18. Moq - 4.18.4

Test

Il codice sorgente verrà, per quanto possibile (vedi dettagli nella sezione Tempistiche), scritto con l'approccio Test Driven Development, pertanto ogni untà avrà la sua classe di testing. Il pacchetto utilizzato per fare i test è XUnit.

Tempistiche

Di seguito le fasi di lavoro e le relative tempistiche stimate dal team di sviluppo per realizzare la il software per la Stazione Concordia:

1. **fase di progettazione** (4 giorni)
    - Definire il design architetturale
    - Studiare e testare la API di Trello
    - Definire e creare Database tramite Entity Framework (code first)

2. f**ase realizzazione livello persistence** (1 giorno)
    - Definizione e scrittura metodi Repository (TDD)
3. **fase realizzazione livello business logic** (8 giorni)
    - Creazione delle classi DTO
    - Automapper
    - Definizione e scrittura metodi Controller
    - Creazione classe per verificare la presenza di connessione (TDD)
    - Creazione dei Servizi (TDD)
    - API consumer (TDD)
4. **fase realizzazione livello presentation** (4 giorni)
    - Creazione interfaccia web per scienziati base
    - Controllers
5. **fase test pre-release** (1 giorni)
    - Fase di test prima della messa in produzione (utilizzo utente)
    - Rilascio applicazione