CONCORDIA STATION

**Abstract**

Implementare un applicativo web che permetta di ottenere una dashboard dei task aggiornata in base al periodo di tempo di connessione ad internet delle basi scientifiche, dove si svolgono gli esperimenti.

I task riguardano lavori che uno scienziato deve svolgere. Ogni task avrà una priorità e uno status (in svolgimento, non svolto o terminato).

Gli scienziati che accedono alla dashboard visualizzano i task in ordine di priorità e interagiscono aggiungendo commenti ai task, modificano lo status del task e la percentuale di svolgimento del task si aggiorna ad ogni interazione tra scienziato task.

**Requisiti**

* *Mostrare agli scienziati un’interfaccia che mostri loro gli esperimenti e le fasi di cui si compongono*
* *Gli scienziati lavorano ai task assegnati*
* *Allineamento centri di ricerca, ovvero rendere possibile la migrazione dei dati da una Api Rest ad un database locale*
* Permettere agli scienziati di aggiungere commenti alle fasi dell’esperimento e cambiare lo status di una specifica fase
* *Parametrizzare i valori con i quali viene l’allineamento poiché l’orario e la finestra di tempo di passaggio del satellite, e quindi della connessione con il resto del mondo*
* *Definire già i valori del punto precedente per poter coprire tutto l’anno solare senza mettere mano al codice una volta deployato*

**Contratti REST di tutti i servizi utilizzati**

* **Metodo Get; URL:** [**https://api/**](https://api/)**Login**

**Verificare la chiamata di tipo Get all’applicativo per accedere con le credenziali e visualizzare** [**https://api/index**](https://api/index) **dove si vedranno tutti gli esperimenti**

* **Metodo Get; URL:** [**https://api/Phase/Details**](https://api/Phase/Details)

**Verificare la chiamata di tipo Get ai dettagli di una specifica fase dell’ esperimento visibile nell’interfaccia**

* **Metodo Put; URL:** [**https://api/Index/Details**](https://api/Index/Details)**?phaseId=”Idphase”**

**Verificare se la chiamata di tipo Put per aggiungere un commento all’interno della pagina di visualizzazioni di commenti per fase**

* **Metodo Put;** [**https://api/Phase/UpdateStatus**](https://api/Phase/UpdateStatus)

**Verificare la chiamata per modificare lo status di una specifica fase dell’esperimento**

**SOLUZIONE PROPOSTA**

**Immagine che contiene schermata, bianco, nero, Elementi grafici

Descrizione generata automaticamente**

“ConcordiaStation” rappresenta il cuore dell’applicazione dove sono gestite tutte le logiche dei requisiti, sono strutturati i modelli per l’interazione con il database locale e per la creazione dell’interfaccia grafica.

“Localdb” è il database dove verranno salvati gli esperimenti, le fasi degli esperimenti, i commenti e gli scienziati che lavorano su di esse. Questa componente sarà interrogata dalle interfacce “repository”, che compongono l’applicativo per ottenere i dati desiderati sfruttando la tecnologia “Entity Framework”.

“SyncConcordiaStation” è una parte dell’applicativo che si occupa di sincronizzare i dati presenti nell’API di Trello al database locale e viceversa, per poter visualizzare i dati più recenti non appena ci si connette ad internet. **Immagine che contiene schermata, design

Descrizione generata automaticamente**

**TECNOLOGIE**

* .NET 6.0
* Entity Framework (ORM)
* Model-View-Controller pattern
* xUnitTest
* SQL Server
* MailKit
* TrelloDotNet

**TEST**

L’applicazione verrà testata tramite UnitTest sui metodi presenti nella classe “Service” di ogni entità. La scelta ricade sul fatto che le classi prese in esame riguardano logiche di business dell’applicativo. Rispettivamente i metodi presi in esame sono:

* GetLastActivity(Comment comment)
* CheckEmail(string email)
* GetIdByEmail(string email)
* CheckPassword(string password)
* AddPhase(int idExperiment, Phase phaseToAdd)
* OrderPhaseByPriority(Experiment experiment)
* UpdatePriorityOnDeadline(Experiment experiment)
* ShowTitle(Experiment experiment)
* ShowDetails(Phase phase)
* AddComment(Comment comment)
* ChangeStatusPhase(int phaseId, Status status)

**TEMPISTICHE**

* Repository e Service delle entità: 1/06/2023
* Test su classi Sevices: 05/06/2023
* Sincronizzazione dei dati: 10/06/2023
* Scrittura del report: 15/06/2023
* Interfaccia web: 17/06/2023
* Eventuali test aggiuntivi e validazione: 21/06/2023
* Deploy: 22/06/2023