

MODERAZIONE CONTENUTI INAPPROPRIATI

CENSURA MESSAGGI IN UNA CHAT

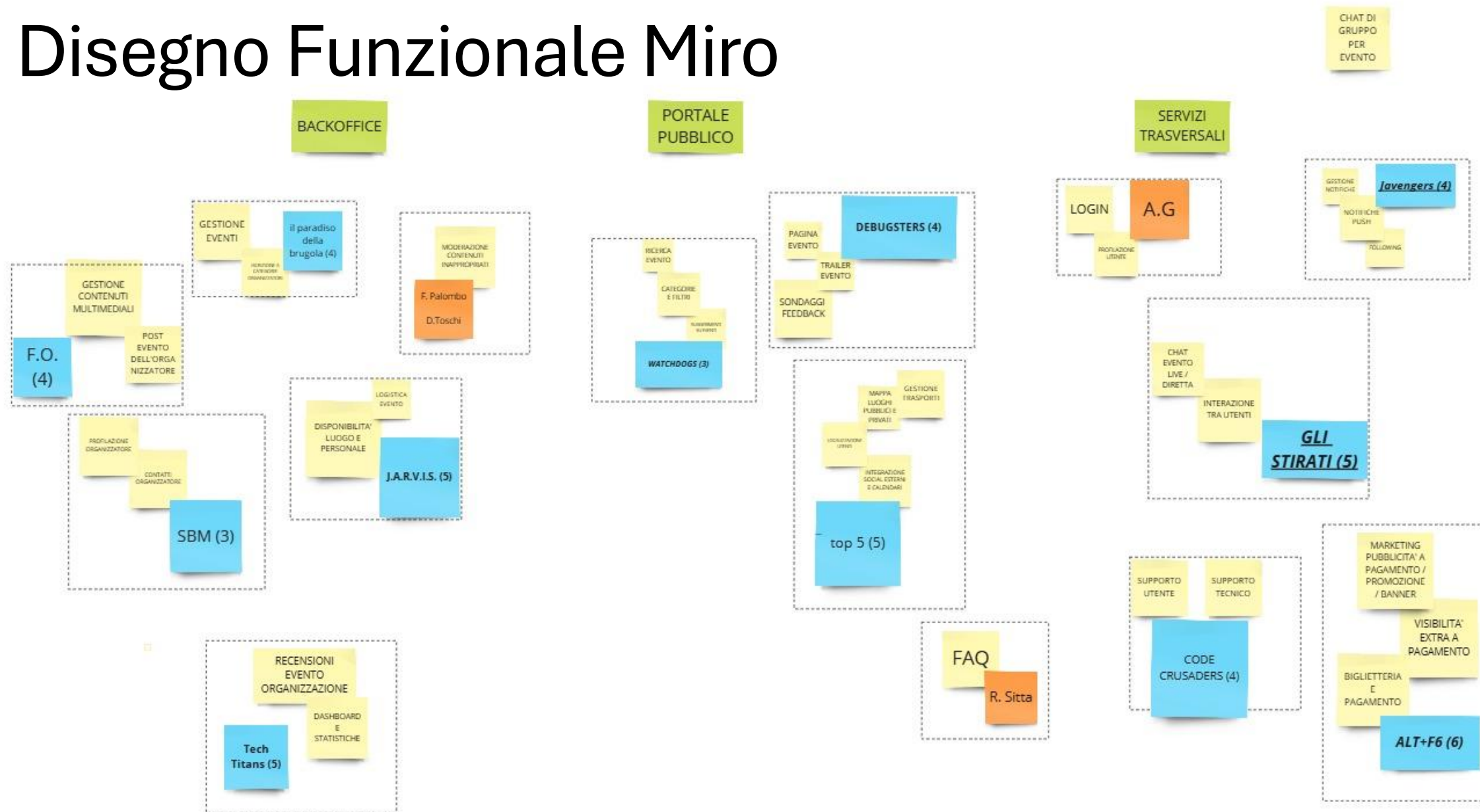
Sistema di Moderazione Chat

- Sviluppo di un sistema backend (e frontend a scopo dimostrativo) per una chat moderata.
- Elaborazione messaggi inviati
- Censura di eventuali parole inadeguate
- Utilizzo di un dataset modulabile di parole bannate

Distribuzione del Team

- Backend Developer: Palombo Francesco
- Frontend Developer: Palombo Francesco
- Tester: Palombo Francesco

Disegno Funzionale Miro



Organizzazione su Trello

The image shows a Trello board interface with a purple header and background. The board is titled "Moderazione Contenuti" and includes navigation options like "Visibile allo Spazio di lavoro", "Bacheca", and "Tabella". It is organized into four vertical lists, each with a title, a move icon, and a more options icon.

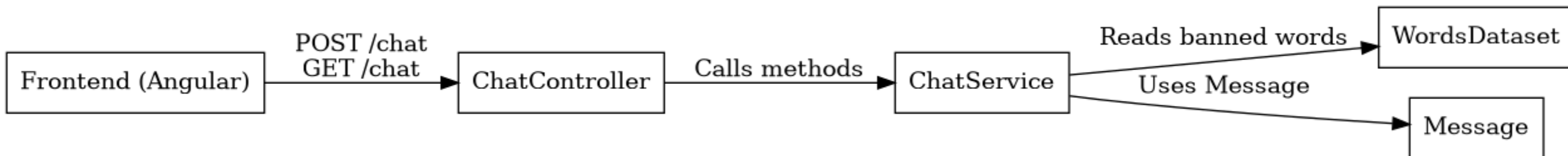
- What Happens?**
 - Un utente invia un messaggio su una chat testuale dal frontend
 - Il frontend invia tale messaggio al backend per la moderazione dei contenuti
 - Ricerca dei termini usati nel messaggio usando un data set di slurs
 - Il backend rimanda a frontend il messaggio con le censure appropriate
 - + Aggiungi una scheda
- What I need?**
 - BannedWords Dataset
 - Oggetto Messaggio per lo scambio di informazioni
 - ChatController per gestire le richieste
 - ChatService che contiene la parte logica dietro la gestione delle richieste
 - + Aggiungi una scheda
- Problemi**
 - [Risolto] Permessi negati CORS
 - [Risolto] Invio dati da Frontend a Backend
 - [Risolto] Imparare Springboot
 - + Aggiungi una scheda
- Considerazioni**
 - Non esiste un miglior metodo per moderare i contenuti se non la moderazione umana. Quindi è importante una funzione per segnalare comportamenti scorretti così la community può aiutare con la moderazione
 - + Aggiungi una scheda

Test cases

ID	Titolo	Dati di Input	Risultato Atteso	Esito
TC001	Verifica messaggio senza parole bannate	«Hello world»	«Hello world»	Pass
TC002	Verifica moderazione parola bannata	«Hello bannedWord world»	«Hello *** world»	Pass
TC003	Verifica invio messaggio nullo	null	Nessun messaggio aggiunto, gestione errore con output di log errore su console.	Pass

Architettura del Progetto

- Pattern MVC: Model-View-Control:
 - **Controller (ChatController)**: Gestisce le richieste HTTP e le mappa sui servizi.
 - **Service (ChatService)**: Logica applicativa per moderare i messaggi e gestire i dati.
 - **Model (Message, WordsDataset)**: Definizione delle entità utilizzate.
- Frontend: Angular per la gestione dell'interfaccia utente
- Comunicazione: API REST tra Angular e Springboot



Funzionalità Implementate

1. Invio di messaggi:
 - Endpoint POST /chat per aggiungere un messaggio
2. Recupero di messaggi:
 - Endpoint GET /chat per ottenere tutti i messaggi inviati
3. Moderazione dei messaggi:
 - Censura delle parole contenute nel dataset

Funzionalità: Censura delle parole bannate

- Richiesta di moderazione (.setMessage) del messaggio ricevuto nell'endpoint POST
- Il Service invocato richiede una Ricerca Binaria sul Dataset
- Ad ogni riscontro positivo: sostituzione parola bannata con '***'
- Salvataggio del messaggio moderato in una lista di cronologia.
- Richiesta di GET da parte del Frontend per mostrare i messaggi inviati

Considerazioni

- Non esiste un metodo ottimale di moderazione dei contenuti se non il controllo umano.

- Non dobbiamo dimenticare la vicenda di
«**The Untold History of Toontown's SpeedChat**»

(Dove Disney nel 1996 volle creare un gioco sicuro per i bambini con un filtro di chat testuale:

“No kid will be harassed, even if they don't know they are being harassed.”).

"We thought it was the perfect solution, until we set our first 14-year old boy down in front of it.

Within minutes he'd created the following sentence: “

I want to stick my long-necked Giraffe up your fluffy white bunny.”)

<http://habitchronicles.com/2007/03/the-untold-history-of-toontowns-speedchat-or-blockchattm-from-disney-finally-arrives/>