SUPPORTO TECNICO E SUPPORTO UTENTE

CodeCrusaders
Antonio D'Arenzo
Luca Ferrari
Matteo Mengoli
Matteo Tosi

DISTRIBUZIONE DEL TEAM

Analista

Antonio D'Arenzo

Architetto

Luca Ferrari

Front End

Matteo Mengoli

Back End

Matteo Tosi

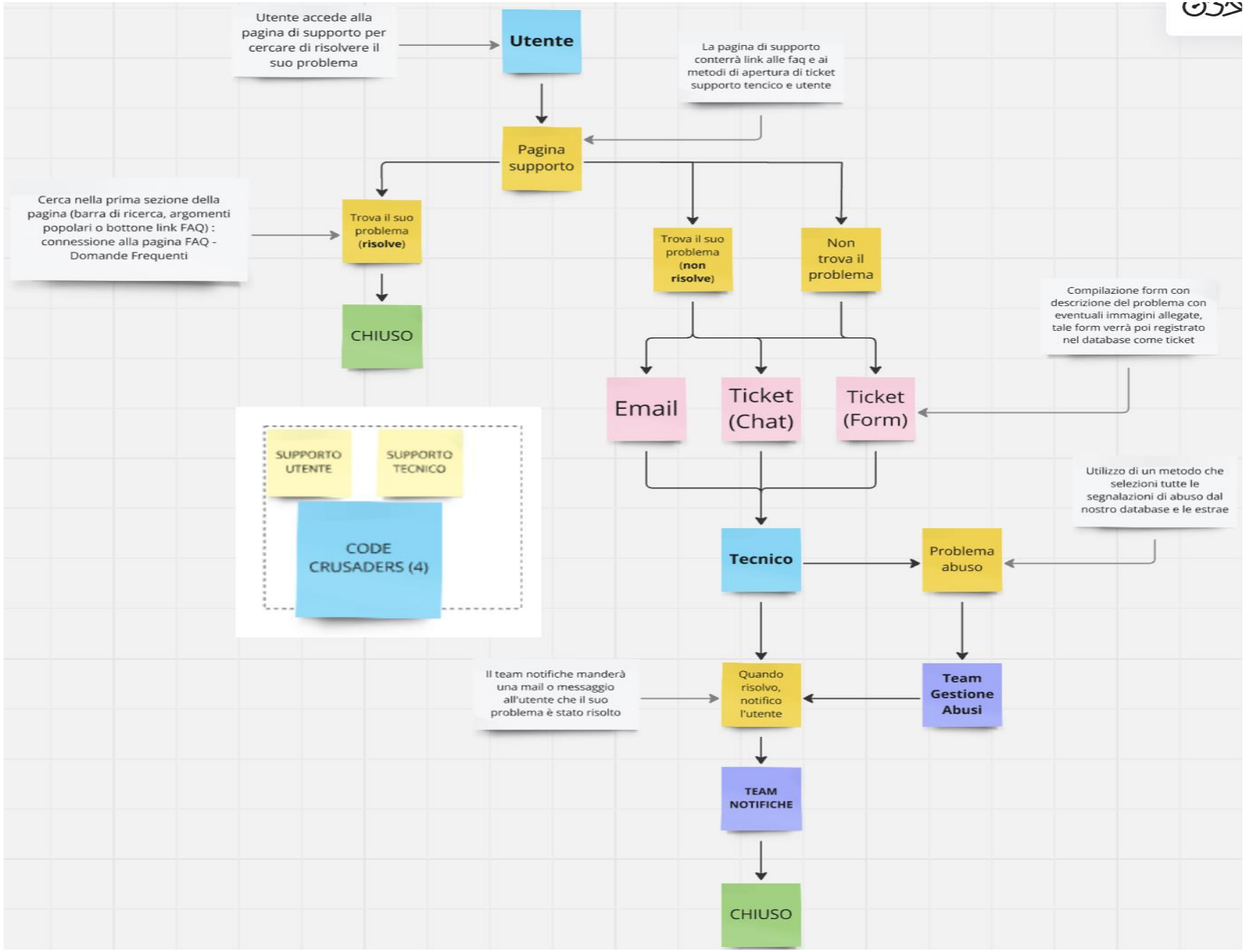
EVIDENZE DELLE CERIMONIE

Sprint Planning: Ogni Lunedì ciascun componente del team pianificava il lavoro da compiere in un intervallo di tempo fissato. Durante la riunione si fissavano gli obiettivi da raggiungere durante lo sprint e lo sforzo necessario per raggiungerli. Alla fine della riunione ciascuno di noi doveva essere in grado di lavorare con autonomia. (Durata '45 minuti).

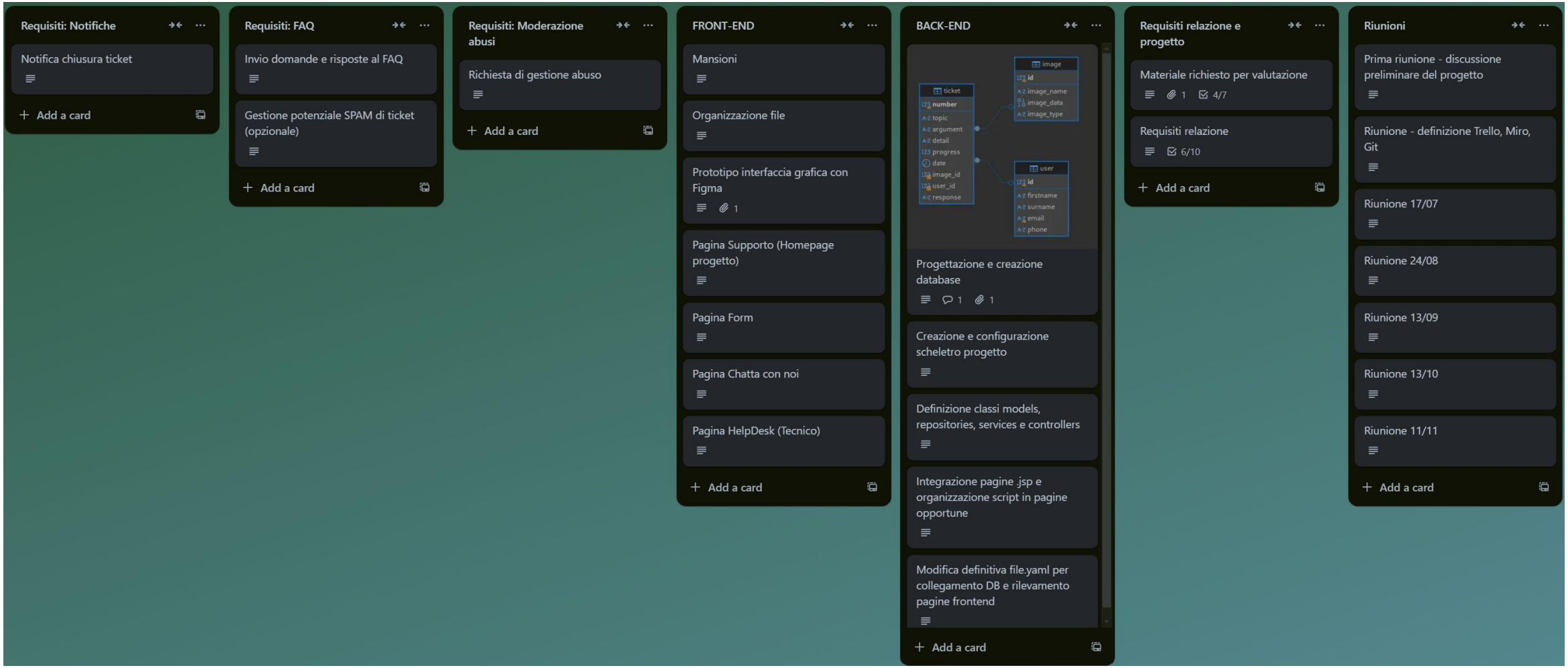
Riunione Giornaliera: Si discuteva via chat Whatsapp del procedimento del progetto e in particolare dei problemi che sorgevano, solo a scopo informativo (Durata '5 minuti)

Sprint Rewiew: Ogni mese facevamo una riunione sull'applicazione discord, dove discutevamo del lavoro effettuato nello sprint. La riunione consisteva nel dare un feedback al lavoro degli altri e discutere sugli aspetti che si potevamo migliorare. Se un membro del team non riusciva a completare il lavoro assegnato, questo veniva redistribuito agli altri membri e la pianificazione veniva adattata di conseguenza. (Durata 2 ore)

DISEGNO FUNZIONALE



TRELLO


Requisiti: Notifiche
Notifica chiusura ticket
Add a card
Requisiti: FAQ
Invio domande e risposte al FAQ
Gestione potenziale SPAM di ticket (opzionale)
Add a card
Requisiti: Moderazione abusi
Richiesta di gestione abuso
Add a card
FRONT-END
Mansioni
Organizzazione file
Prototipo interfaccia grafica con Figma
1
Pagina Supporto (Homepage progetto)
Pagina Form
Pagina Chatta con noi
Pagina HelpDesk (Tecnico)
Add a card
BACK-END
ticket
number
topic
argument
detail
progress
date
image_id
user_id
response
image
id
image_name
image_data
image_type
user
id
firstname
surname
email
phone
Progettazione e creazione database
1
1
Creazione e configurazione scheletro progetto
Definizione classi models, repositories, services e controllers
Integrazione pagine .jsp e organizzazione script in pagine opportune
Modifica definitiva file.yaml per collegamento DB e rilevamento pagine frontend
Add a card
Requisiti relazione e progetto
Materiale richiesto per valutazione
1
4/7
Requisiti relazione
6/10
Add a card
Riunioni
Prima riunione - discussione preliminare del progetto
Riunione - definizione Trello, Miro, Git
Riunione 17/07
Riunione 24/08
Riunione 13/09
Riunione 13/10
Riunione 11/11
Add a card

TEST CASES

Test per garantire che l'utente associato al ticket venga recuperato dal database :

```
// Generated by Qodo Genimport org.junit.Test;
import static org.junit.Assert.*;
public class AddnewticketTest {
// Ensures that the user associated with the ticket is retrieved from the database.
@Test
public void test_user_association_with_ticket() {
// Arrange
 UserService userService = mock(UserService.class);
TicketService ticketService = mock(TicketService.class);
HttpServletRequest request = mock(HttpServletRequest.class);
TicketController ticketController = new TicketController(userService, ticketService);
User existingUser = new User(1, "Jane", "Doe", "jane.doe@example.com", "0987654321", null);
when(userService.getUserByEmail("jane.doe@example.com")).thenReturn(existingUser);
// Act
RedirectView result = ticketController.addNewTicket("Jane", "Doe", "jane.doe@example.com", "0987654321", "Support", "Technical Issue", "Details about
the issue", request);
 // Assert
verify(userService, times(1)).getUserByEmail("jane.doe@example.com");
verify(ticketService, times(1)).addTicket(any(Ticket.class));
assertNotNull(result);
}
}
```

DISEGNO ARCHITETTURALE

Per il nostro progetto abbiamo deciso di adottare una struttura monolitica (JSP con Springboot), in quanto esso è relativamente piccolo ed essendo un'unica entità è più facile da sviluppare e offre performance migliori. Riconosciamo che se l'applicazione dovesse crescere significativamente dovremmo passare ad una architettura a due livelli.

ELENCO FUNZIONALITÀ

Lato Utente:

Contatto servizio supporto tramite:

- servizio di posta elettronica;
- apertura ticket (form);
- chat real-time (Beta da sviluppare ulteriormente).

Lato Tecnico:

Creazione di ticket personalizzati;

Visualizzazione dei ticket e rispettivi dati, con possibilità di filtrarli per stato;

Gestione dei ticket, con possibilità di cambiare risposta inviata e stato.

DETTAGLIO DI APERTURA TICKET TRAMITE FORM

Se l'utente non riesce a risolvere il proprio problema tramite le FAQ, potrà aprire un ticket nella pagina di supporto dedicata. Qui troverà un form in cui inserire le proprie informazioni personali, l'email associata all'account, il numero di telefono e i dettagli relativi al problema, come tematica, argomento, descrizione, ed eventualmente potrà allegare un'immagine.

Quando l'utente invierà il form, tale ticket verrà registrato nella base di dati tramite la funzione java AddNewTicket. Se l'utente esiste già, tale ticket verrà collegato alla mail già registrata sul database altrimenti verrà registrato anche il nuovo utente.

Tale ticket poi potrà essere estratto nella pagina helpdesk dove un tecnico potrà risolverlo.

FEEDBACK ESPERIENZA

Il feedback è complessivamente positivo, questo progetto ci ha mostrato e fatto lavorare con mano a tutte quelle componenti che ci sono state insegnate durante i corsi universitari. Non è stato facile trovarsi faccia a faccia con lo sviluppo di un applicazione in quanto è stato difficile trasmutare le competenze teoriche in competenze pratiche.

Il lavoro di gruppo è stato molto impegnativo, in un contesto universitario ogni membro ha diversi tempi e diverse priorità di lavoro. Abbiamo trovato non poche difficoltà nel sincronizzare i lavori e molte volte abbiamo avuto idee contrastanti.