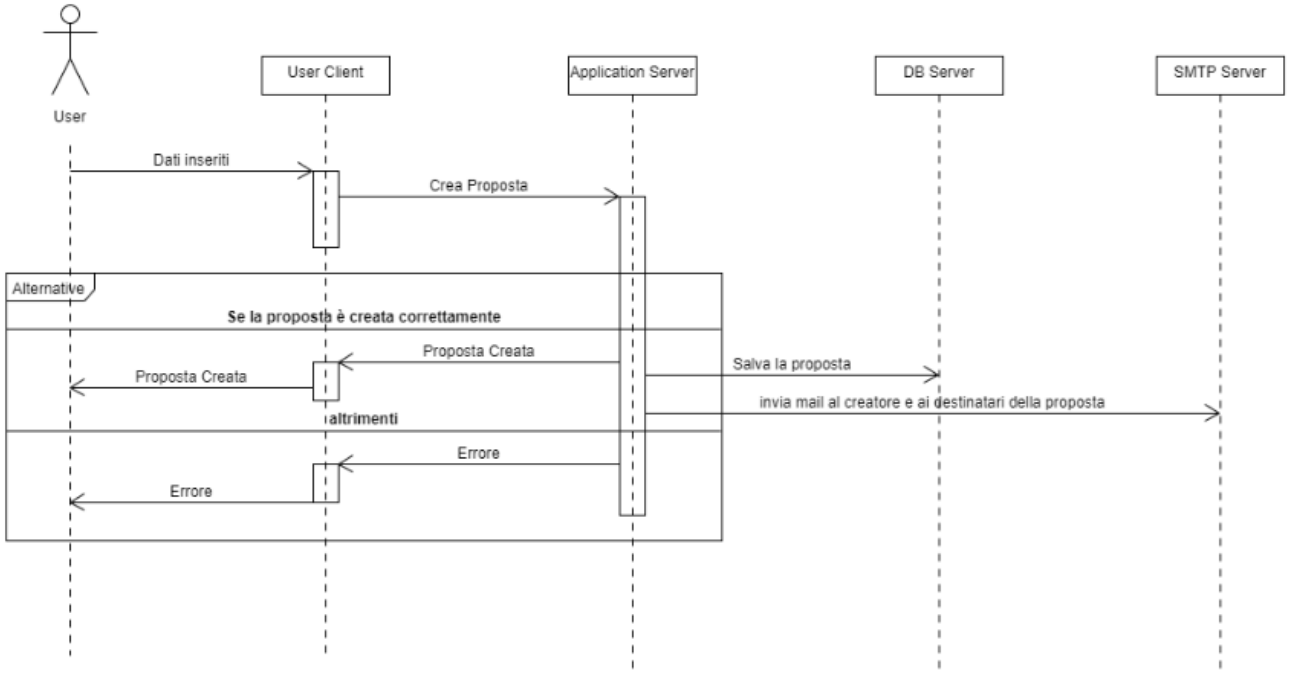
Iterazione 1

Come primo passo, osservando il Sequence Diagram prodotto nella fase di Envisioning e i valori di priorità dati ai casi d'uso, decido di approfondire e iniziare a realizzare il caso d'uso UC1 e parte del caso d'uso UC2.

Nome	**ID**	**Tipo**	**Priorità**	**Descrizione**	**Req. padre**	**Req. figli**
CREA PROPOSTA	UC1	funzionale	3	Deve permettere all'utente di creare una proposta di transazione		UC2
GENERA MAIL AUTOMATICHE	UC2	funzionale	3	Il sistema deve inviare agli utenti email con informazioni riguardanti lo stato di una transazione/registrazione	UC1, UC3, UC4, UC5	

Approfondisco inoltre il Sequence Diagram ad essi collegato.



Prima di iniziare a scrivere codice, applico i **design heuristic steps**.

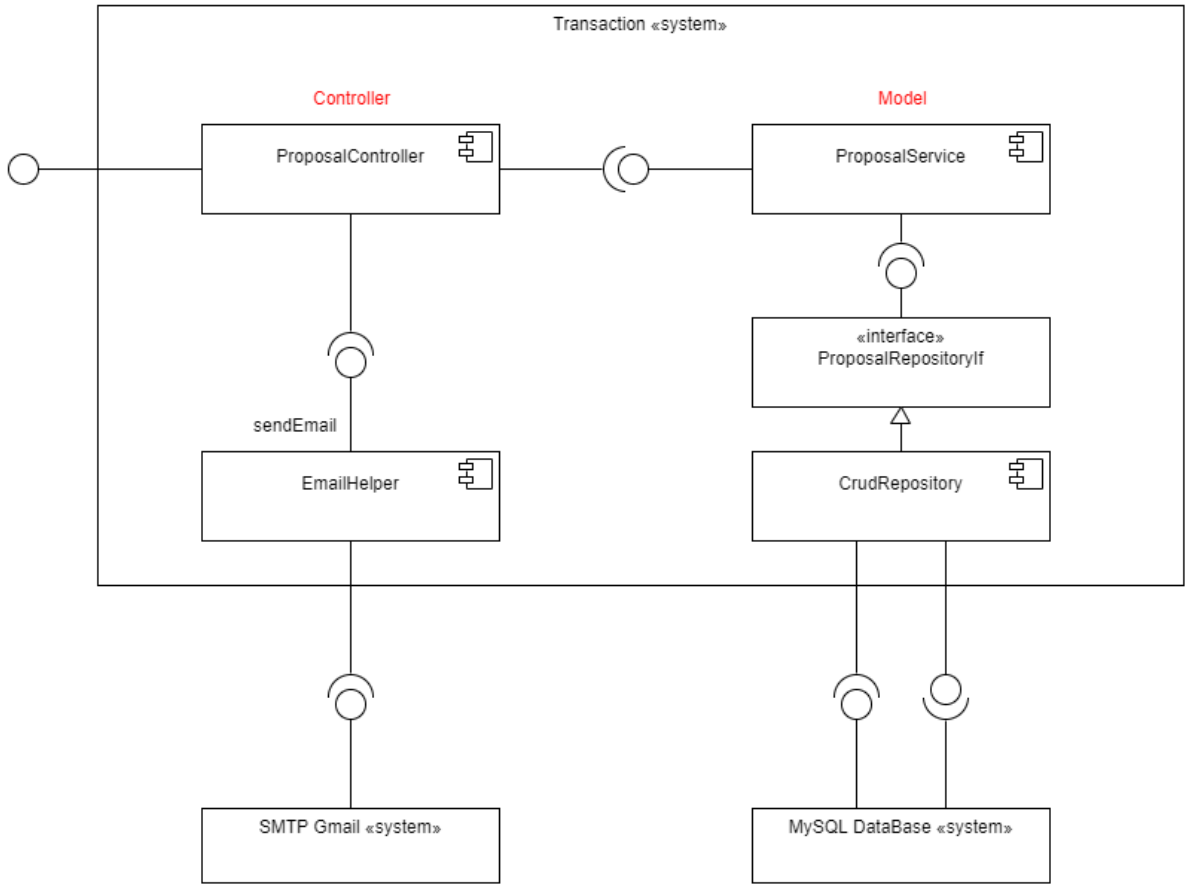
Il caso d'uso prevede la creazione di una proposta: l'utente inserisce i dati che compongono una proposta. In prima battuta, e per semplicità, considero tre dati: *codice identificativo*,

nome e una breve stringa di testo (*descrizione*). Serve una classe Proposal che rappresenti la struttura dati.

Prevedo poi la presenza di *ProposalController*: componente software che si occupa di gestire le chiamate HTTP riferite all'oggetto Proposal. Di conseguenza, questo oggetto si occupa di esporre le API.

Inoltre, questi dati devono essere salvati in modo persistente su un database, dunque creo un componente *ProposalService* che si occupa della gestione del salvataggio dei dati.

Di fatto sto applicando il pattern **Model-View-Controller**. Il componente View è simulato tramite l'uso di Postman, che mi permette di testare gli endpoint del Controller.



Per la scrittura del codice scelgo Java e per la gestione delle API mi affido al framework SpringBoot, che fornisce interfacce e librerie predisposte per la scrittura di codice api e web oriented.

Le interfacce fornite da ProposalController, utilizzabili dalla View sono:

GET	getAllProposals	Ritorna il vettore di tutte le Proposal presenti nel DB.
POST	createProposal	Crea una nuova Proposal. Se esiste già o c'è qualche errore nei dati viene sollevata un'eccezione dal sistema e ritornato un messaggio di errore come response.

UPDATE	updateProposal	Aggiorna i dati di una proposal, se l'id associato ad un Proposal non esiste, non è possibile aggiornare: viene tornato un messaggio di errore
DELETE	deleteProposal	Elimina una Proposal. Se l'oggetto non esiste, errore.

In questa iterazione il sistema è interamente eseguito in locale a bordo del mio notebook. Mediante il framework Spring viene mandata in esecuzione una istanza di Tomcat, sul quale viene fatto girare il codice da me scritto.

Sempre in locale, ho installato MySQL e, tramite il software MySQL Workbench, gestisco il database a livello di status ed eseguendo query.

Inoltre, tramite l'interfaccia JavaMailSender, sono in grado di comunicare con il server remoto SMTP di GMail, tramite il quale sono in grado di inviare e-mail. A questo scopo ho creato una apposita casella agalliani.test@gmail.com.

Nelle application.properties, oltre alle proprietà di gestione del database, ho dovuto inserire le configurazioni necessarie per la gestione delle mail:

```
## Email Properties
spring.mail.host = smtp.gmail.com
spring.mail.port = 587
spring.mail.username = agalliani.test@gmail.com
spring.mail.password = ***********

spring.mail.properties.mail.smtp.auth = true
spring.mail.properties..mail.smtp.starttls.enable = true
```