Studio Medico Gruppo 2 Java 8

Obbiettivi

Dare la possibilità di effettuare delle prenotazioni per n studi medici. Tramite il linguaggio Java ed il framework Spring Boot interagiremo con delle chiamate API per dare la possibilità ad un paziente di prenotare presso uno studio medico e salvare la prenotazione in un database.

Attori Coinvolti

Un medico, il paziente, il segretario, la prenotazione.
Medico: deve poter visualizzare le prenotazioni.
Paziente: può prenotare, visualizzare solo la sua prenotazione e disdirla.
Segretario: può inserire prenotazioni, visualizzare ed eliminare e modificare, cioè può fare tutte le operazioni CRUD.

Relazioni tra gli attori:

- Medico 1vsN Paziente
- Medico 1vs1 Segretario
- Medico 1vsN Prenotazione
- Paziente 1vs1 Prenotazione

Requisiti Funzionali

- Sistema di prenotazione degli appuntamenti: Il software deve consentire ai pazienti di prenotare gli appuntamenti in modo rapido e semplice, scegliendo il medico o la specialità richiesta, la data e l'ora dell'appuntamento.
- Gestione delle cancellazioni: Il sistema di prenotazione deve consentire ai pazienti di cancellare gli appuntamenti prenotati, nel

caso in cui ci sia un imprevisto. Il software deve anche inviare una notifica al personale medico in caso di cancellazione.

- Accesso ai dati del paziente: Il software deve consentire al personale medico di accedere rapidamente alle informazioni sui pazienti, come la loro storia clinica, i farmaci prescritti e le note del medico.

Tecnologie utilizzate:

1. Spring Rest
2. Spring Boot
3. Java
4. JPA
5. SQL
6. Swagger

Analisi del sistema

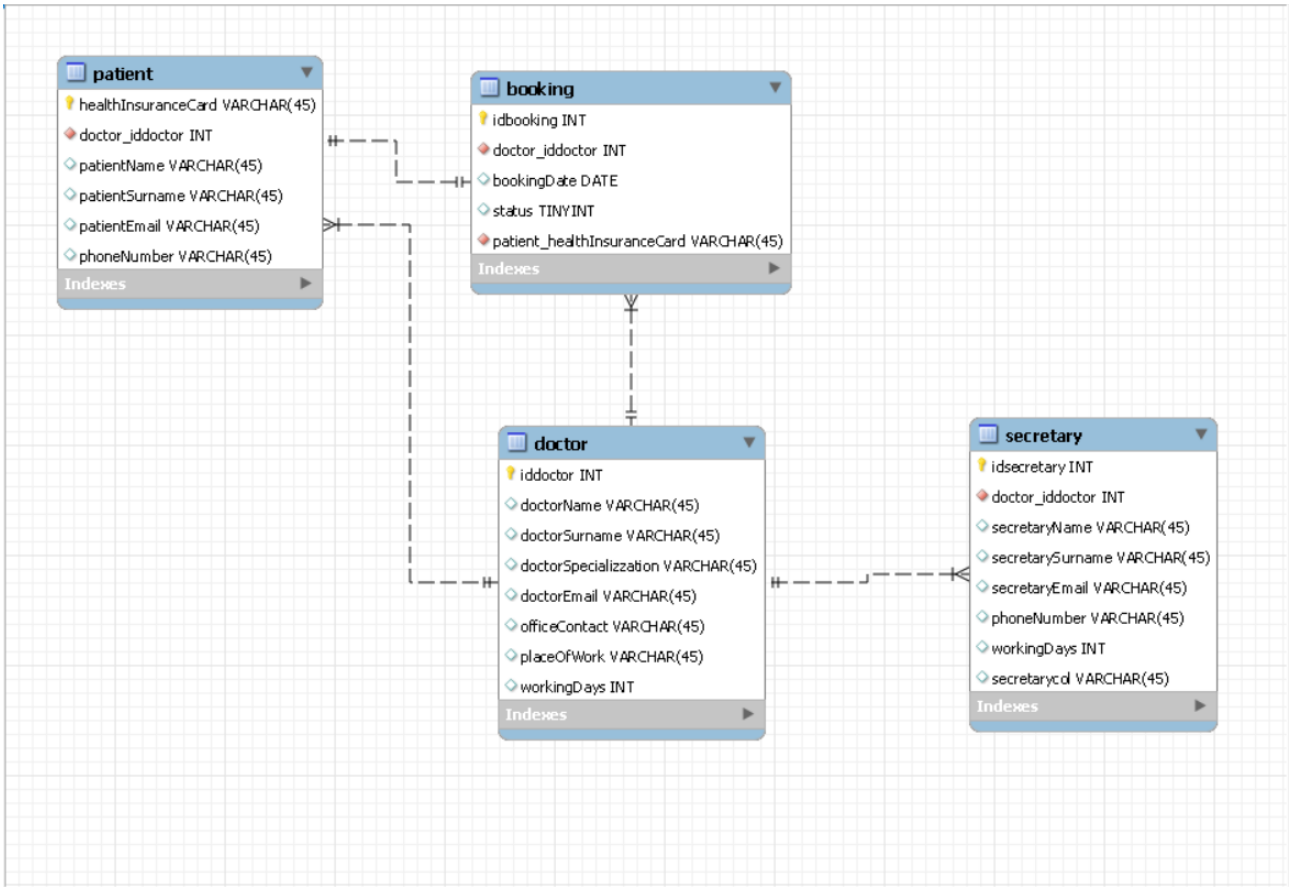
Il medico è caratterizzato da un'id, il nome, il cognome, l'e-mail, il numero di telefono dell'ufficio, la specializzazione, la località in cui lavora e il numero di giorni lavorativi. Il controller del medico avrà dunque un metodo di POST che permette la creazione di un nuovo medico ed un metodo di GET che permette la visualizzazione di tutti i medici presenti nel database.

Il paziente è caratterizzato da un'id, il nome, il cognome, l'e-mail, il numero di telefono e il codice fiscale. Il controller del paziente avrà dunque un metodo di POST che permette la creazione di un nuovo paziente ed un metodo di GET che permette la visualizzazione di tutti i pazienti presenti nel database.

Il segretario è caratterizzato da un'id, il nome, il cognome, l'e-mail, il numero di telefono e il numero dei giorni lavorativi. Il controller del segretario avrà dunque un metodo di POST che permette la creazione di un nuovo segretario ed un metodo di GET che permette la visualizzazione di tutti i segretari presenti nel database.

La prenotazione è caratterizzata dalla data di creazione e lo status. Il controller della prenotazione avrà dunque un metodo di POST che permette la creazione di una nuova prenotazione, un metodo di GET che permette la visualizzazione di tutte le prenotazioni presenti nel database, un metodo di GET che permette tramite ID di visualizzare la singola prenotazione, un metodo di PUT che permette la modifica dei dati della prenotazione, un metodo di DELETE che permette l'eliminazione tramite ID della singola prenotazione ed infine un metodo di DELETE che permette la cancellazione di tutte le prenotazioni presenti nel database.

Diagramma ER



Idee per soluzioni

Creare un progetto tramite Spring initializer con le dovute dipendenze analizzate sopra, creare delle classi entità che avranno all'interno le informazioni di ogni singolo attore in gioco, creare dei controller che hanno i metodi richiesti per esporre le entità al database, creare delle repository per ogni entità in modo da implementare i metodi della JPA repository all'interno dei controller, creare delle classi di service per poter mettere in correlazione le entità coi metodi del controller di altre entità.

Testare le chiamate di ogni controller tramite Swagger-UI e/o Postman per verificare la corretta funzionalità di ogni endpoint.

Utilizzare infine Swagger-UI per documentare l'intero progetto con l'utilizzo di SpringDoc.