Jag valde MongoDB som databas för applikationen eftersom jag redan hade använt PostgreSQL i backendutveckling del 1. För just detta projekt spelar valet av databas egentligen ingen större roll, då det är av mindre skala. Om jag däremot skulle behöva välja mellan PostgreSQL och MongoDB för ett framtida projekt, skulle jag välja MongoDB. Anledningen är dess flexibla, dokumentorienterade lagring i form av Binär JSON (BSON) samt den smidiga integrationen med Node.js.

Tekniker och npm-paket

- **Node.js** är den absolut viktigaste tekniken i denna applikation, vilket möjliggör körning av programkod utanför webbläsaren.

- **Express.js** är ett minimalt och flexibelt NPM paket som underlättar hanteringen av nätverksförfrågningar på serversidan. Det ansvarar för applikationens API-förfrågningar i form av Create, Read, Update och Delete.

- **TypeScript** är ett programmeringsspråk som har varit mycket användbart för att fördefiniera typer för flera Mongoose-scheman och andra delar av applikationen. TypeScript lägger till statiska typer till JavaScript, vilket har resulterat i färre buggar i koden. Dess felhantering fångar potentiella fel under utvecklingsfasen.

- **Mongoose** är ett Object Data Modeling (ODM) bibliotek som underlättar utvecklingen med MongoDB och Node.js. För att Mongoose ska fungera korrekt krävs att ett schema införs, vilket definierar strukturer och regler för databasen, även om MongoDB i sig inte har något sådant krav. I Trullo finns det tre scheman: User, Task och Project. Dessa används för att utföra CRUD-operationer med Mongoose inbyggda metoder.

- **Bcrypt** är ett paket som används i applikationen för att kryptera känslig data innan den skickas och lagras i databasen. I mitt fall används Bcrypt för att kryptera lösenordet samt "secret key" som används för att återställa lösenordet.

- **JWT** (JSON web token) är ett paket som används för autentisering och auktorisering i Trullo.

Hur Applikationen fungerar

Applikationen är strukturerad som ett RESTful API med flera endpoints för CRUD-operationer. Dessa operationer hanterar information som skickas mellan servern och databasen när det gäller projekt, användare och uppgifter. Med hjälp av JWT-tokens tilldelas varje användare en roll (user eller admin), vilket medför olika behörigheter. En användare med rollen admin har dessutom extra behörighet att ändra alla användaruppgifter i databasen.

Varje uppgift (task) är kopplad till ett projekt, som i sin tur innehåller en lista över alla uppgifter relaterade till projektet. När en task skapas, uppdateras eller tas bort, uppdateras även projektet.