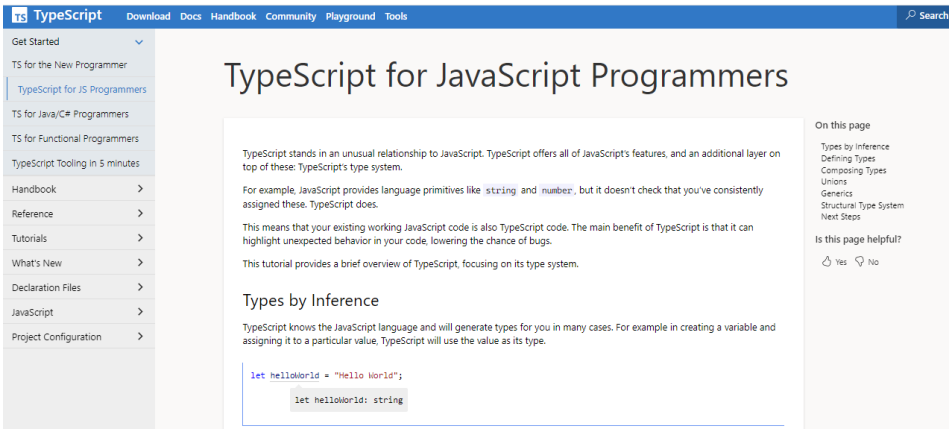
BAB V
HASIL DAN PEMBAHASAN

Dalam proses pengembangan aplikasi, terdapat beberapa tahapan yang harus dilakukan untuk mencapai hasil yang optimal. Tahapan-tahapan tersebut meliputi research, brainstorming, desain UI/UX, coding, troubleshoot, testing dan proses deployment. Berikut merupakan tahapan yang dilalui dalam masa Praktik Kerja Lapangan di PT. Indosat Tbk.

5.1 Penyelesaian Kursus Typescript, React, dan Express.js

Pada awal kegiatan, sembari penulis menyiapkan desain untuk UI/UX serta desain *database* penulis diarahkan untuk belajar Typescript karena penggunaan bahasa pemrograman Typescript dikenal atas keamanannya dan kecepatannya. Aspek tersebutlah yang membuat *project* kali ini dikembangkan dengan bahasa Typescript yang juga merupakan superset dari bahasa Javascript. Berikut merupakan laman resmi dokumentasi Typescript.



Gambar 5. 1 Laman dokumentasi resmi typescript

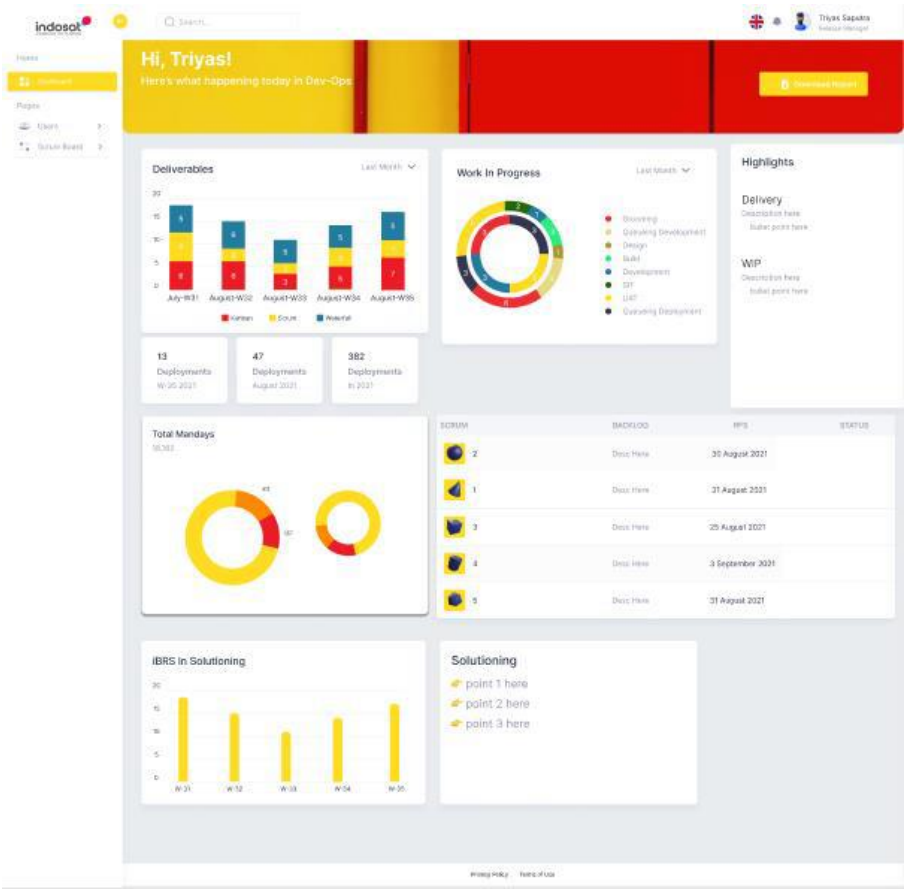
Proses pembelajaran mengenai pengembangan aplikasi dengan Typescript dilakukan dengan beberapa cara yang dapat meningkatkan efektivitas dan efisiensi proses pembelajaran. Salah satu cara yang diterapkan adalah melalui proses brainstorming, yang bertujuan untuk mengumpulkan ide-ide dan konsep pembelajaran yang akan diterapkan. Selain itu, self-learning juga digunakan sebagai metode pembelajaran yang efektif. Self-learning meliputi beberapa cara seperti membaca dokumentasi resmi, membaca dokumentasi komunitas, serta melihat video Youtube terkait pengembangan dengan Typescript.

Membaca dokumentasi resmi merupakan cara yang efektif dalam memahami konsep dan fitur yang ditawarkan oleh Typescript. Dokumentasi resmi dapat diakses melalui website resmi dari Typescript sesuai Gambar 5.1.

5.2 Desain UI/UX

Dalam proyek ini, aspek *user-friendly* sangat diperhatikan untuk mempermudah pengguna dalam memahami data yang disajikan. Oleh karena itu, riset tren dilakukan untuk menentukan desain yang sesuai dengan kebutuhan pengguna. *Software* desain Figma digunakan dalam proses ini untuk membuat desain yang intuitif dan mudah digunakan.

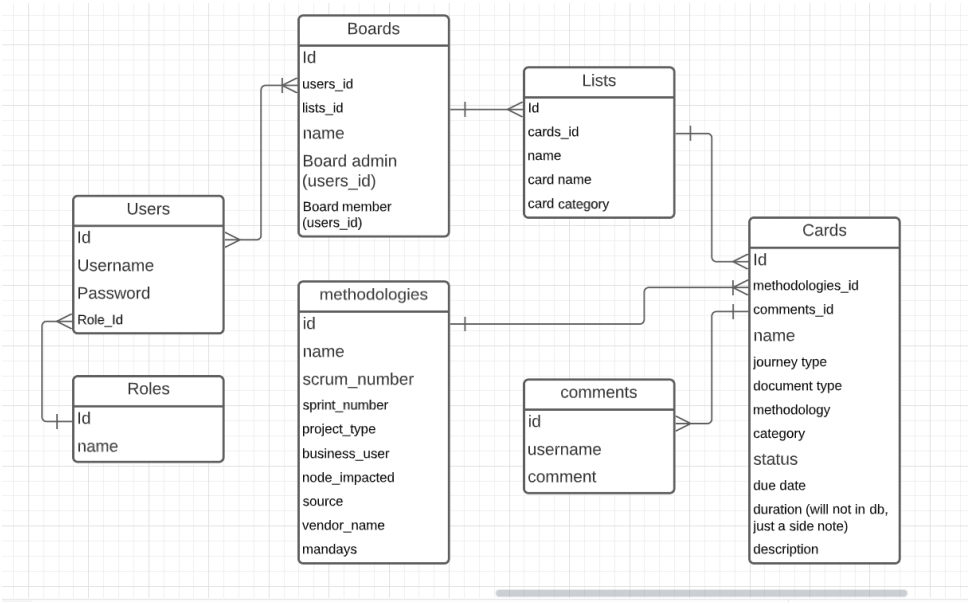
Dengan melakukan riset tren dan menggunakan software desain Figma, serta dengan mendapatkan *feedback* dari para mentor melalui *daily meeting*, diharapkan desain UI dapat sesuai dengan kebutuhan pengguna dan memberikan pengalaman yang baik bagi pengguna dalam mengakses dan memahami data yang disajikan. Berikut merupakan hasil desain *dashboard* projek ini.



Gambar 5. 2 Desain UI Dashboard

5.3 Merancang Entity Relationship Diagram (ERD) Database

Pada tahap ini, dilakukan iterasi pekerjaan untuk perbaikan dan perubahan sesuai dengan *feedback* yang telah diterima sebelumnya. Aspek keamanan dan kemudahan akses data menjadi fokus utama dalam perancangan database untuk aplikasi projek manajemen ini. Hal ini penting untuk memastikan bahwa data yang disimpan dalam aplikasi aman dari akses yang tidak sah dan mudah diakses oleh pengguna yang berwenang. Beberapa teknik seperti enkripsi dan autentikasi digunakan untuk meningkatkan keamanan data. Selain itu, perancangan database juga memperhatikan aspek scalabilitas dan performa untuk memastikan aplikasi dapat berjalan dengan baik dan cepat. Berikut merupakan hasil ERD pada tahapan ini.



Gambar 5. 3 *Entity Relationship Diagram*

Gambar 5.2 diatas merupakan diagram penjelasan dari hubungan antara entitas pada setiap fitur yang akan dibuat.

5.4 Mengembangkan Back-end

5.4.1 Desain *Web Service*

Proses desain sistem web service REST API bertujuan untuk merancang fungsi-fungsi yang akan digunakan sebagai dasar untuk implementasi API dengan menggunakan metode REST. Desain API ini didasarkan pada hasil analisis kebutuhan yang akan digunakan dalam aplikasi *Project Management*. REST

(Representational State Transfer) adalah salah satu metode arsitektur yang digunakan dalam pengembangan *web service*, REST menggunakan protokol HTTP sebagai jembatan komunikasi antara *client* dan *server*. Metode ini memungkinkan implementasi yang fleksibel dan mudah digunakan.

API yang dirancang akan menyediakan fungsi-fungsi yang diperlukan untuk mengakses dan mengelola data yang disimpan dalam aplikasi *Project Management*.

Tabel 5. 2 Desain *Web Service*

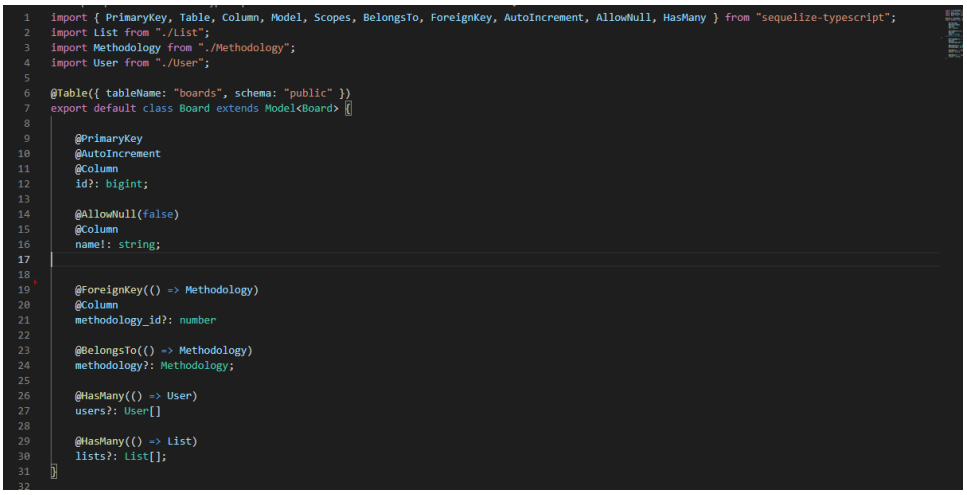
No.	Method	*Endpoint*
1	POST	/users/register
2	POST	/users/login
3	PUT	/users/update
4	DELETE	/users/delete
5	POST	/boards
6	GET	/boards
7	PATCH	/boards/:boardId
8	DELETE	/boards/:boardId
9	POST	/cards
10	GET	/cards
11	PUT	/cards/:cardId
13	PATCH	/cards/update-list/:cardId
14	DELETE	/cards/:cardId

5.4.2 Implementasi Backend

Implementasi REST API untuk *web service* dilakukan dengan menggunakan bahasa pemrograman Typescript. Untuk mengelola permintaan-respon dan routing operasi HTTP, digunakan *framework* Express yang merupakan web framework pada server side. *Framework* ini digunakan untuk memudahkan pengelolaan routing dan pengelolaan *request-response*.

Selanjutnya, *router* dibuat untuk menentukan data yang dikirimkan dan dapat diakses oleh pengguna. *Router* yang dibuat mencakup seluruh *endpoint* API, sehingga efektif dalam pertukaran data antara *client* dan *server*. Dengan menggunakan Typescript sebagai bahasa pemrograman, dapat meningkatkan

keamanan dan stabilitas dari aplikasi yang dikembangkan. Typescript memberikan fitur seperti *strong type checking* yang akan membantu dalam mencegah kesalahan yang mungkin terjadi pada saat *runtime*. Hal ini akan membuat aplikasi lebih aman dan stabil saat digunakan.



```
import { PrimaryKey, Table, Column, Model, Scopes, BelongsTo, ForeignKey, AutoIncrement, AllowNull, HasMany } from "sequelize-typescript";
import List from "./List";
import Methodology from "./Methodology";
import User from "./User";

@Table({ tableName: "boards", schema: "public" })
export default class Board extends Model<Board> {

    @PrimaryKey
    @AutoIncrement
    @Column
    id?: bigint;

    @AllowNull(false)
    @Column
    name!: string;


    @ForeignKey(() => Methodology)
    @Column
    methodology_id?: number

    @BelongsTo(() => Methodology)
    methodology?: Methodology;

    @HasMany(() => User)
    users?: User[]

    @HasMany(() => List)
    lists?: List[];
}
```

Gambar 5. 4 *Source code* Backend

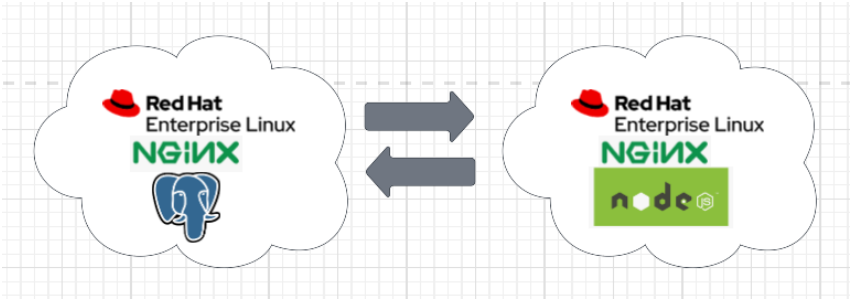
Pada gambar 5.4, terdapat contoh *source code* penginstiantsian model.

5.5 REST API *Service Deployment*

Proses *deployment* aplikasi pada *server* meliputi beberapa tahap penting, yaitu konfigurasi *database*, setup *environment*, dan konfigurasi aplikasi. Konfigurasi *database* merupakan proses penting dalam menyiapkan aplikasi untuk dijalankan pada *server*, termasuk menentukan parameter koneksi ke *database* seperti nama *server*, nama pengguna, dan kata sandi. Selain itu, menyiapkan tabel-tabel dan struktur data juga diperlukan untuk pengembangan aplikasi.

Setup *environment* aplikasi juga merupakan proses penting dalam persiapan aplikasi untuk dijalankan pada *server*. Hal ini termasuk menentukan parameter konfigurasi seperti versi aplikasi, versi Node.js, dan *dependency* yang diperlukan oleh aplikasi. Deployment API *service* pada *server* memerlukan proses yang cukup kompleks, seperti konfigurasi *database*, setup environment, dan konfigurasi aplikasi. Proses ini harus dilakukan dengan benar agar dapat digunakan dengan baik. Melalui pemahaman yang baik tentang sistem operasi

Linux dan *error code* handling, dapat membuat proses *deployment* menjadi lebih efisien dan dapat mengatasi masalah yang mungkin terjadi. Berikut merupakan arsitektur aplikasi yang dikembangkan.



Gambar 5. 5 Arsitektur Aplikasi

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil yang didapat dari tahap perencanaan dan pengimplementasian, dapat disimpulkan bahwa penggunaan REST (Representational State Transfer) API dalam pengembangan *Web Service* mampu meningkatkan efisiensi dalam proses pertukaran data antara *client* dan *server*. Selain itu, penggunaan REST API juga efektif dalam berkomunikasi melalui seluruh endpoint yang telah dibuat.

REST API menyediakan metode yang dapat digunakan untuk mengakses resource pada server melalui operasi HTTP seperti GET, POST, PUT, PATCH, dan DELETE. Hal ini memberikan fleksibilitas dalam proses pengambilan dan pengubahan data. Selain sebagai jembatan komunikasi, REST API juga membantu dalam proses kolaborasi antar sistem dan dapat membantu pengembangan modul yang terkait agar menjadi lebih baik.

Format JSON (JavaScript Object Notation) dipilih sebagai hasil output dari REST API karena format ini merupakan teks sederhana dan lebih mudah diolah. JSON merupakan format yang universal dan dapat digunakan pada berbagai jenis platform. Hal ini memudahkan dalam proses pengembangan aplikasi dan pengolahan data.

6.2 Saran

Untuk meningkatkan hasil penelitian selanjutnya, diperlukan penelitian lanjut untuk mengetahui risiko yang mungkin muncul dari pengembangan skalabilitas untuk penggunaan REST API. Hal ini dikarenakan penggunaan REST API belum tentu cocok untuk sistem yang semakin beasr. Oleh karena itu, perlu dilakukan analisis risiko sebelum melakukan implementasi lanjutan denganREST API.

Selain itu, pengembangan fitur tambahan seperti Redis (Remote Dictionary Server) dapat meningkatkan kinerja modul yang digunakan. Redis merupakan *database* yang dapat digunakan untuk *caching* dan *session management*. Dengan menggunakan Redis, dapat meningkatkan kecepatan aplikasi dan mengurangi beban pada database utama.

Secara keseluruhan, penelitian lanjut yang fokus pada risiko skalabilitas dari penggunaan REST API serta pengembangan fitur tambahan seperti Redis dapat meningkatkan hasil dari penelitian selanjutnya.