Node.js is an open-source, cross-platform, JavaScript runtime environment. It executes JavaScript code outside of a browser.

Prepare

請先安裝 node.js 作為伺服器 與 postgres 作為資料庫, 建議作業系統使用 Ubuntu 18.04 LTS以上或是 Centos 8

```
$ node -v # 檢查 node 是否安裝完成
$ npm  -v # 檢查 npm  是否安裝完成
$ psql -v # 檢查 psql 是否安裝完成
```

最後, 請於專案根目錄執行 `npm install` 將第三方套件安裝, 並執行 `initial.postgre.sql` 以及 `npm run map` 來初始化資料庫, 最後使用 `npm run start` 來開啟伺服器

Project Structure

```
- cert                  #HTTPS憑證
- dist                  #編譯後的檔案
- prisma                #資料庫抽象
- node_modules          #第三方Lib
- spec                  #測試檔案
- src                   #原始檔案
- api.md                #API 文件
- initial.postgre.sql   #資料庫初始化SQL語法
- README.md             #文件
- tsconfig.build.json   #編譯用ts組態
- tsconfig.json         #開發用ts組態
- package.json          #函式庫管理
- yarn.lock             #函式庫詳細清單
```

Database

目前使用 Prisma IDL(Interface Define Language) 來定義資料庫與TypeScript類別的對應,

在最開始的階段, 先使用SQL來決定好資料庫的抽象, 如以下SQL:

```
CREATE TABLE "Users"(
  "user_id"         VARCHAR NOT NULL PRIMARY KEY,
  "user_name"       VARCHAR NOT NULL,
  "created_at"    TIMESTAMP NOT NULL DEFAULT(CURRENT_TIMESTAMP),
  "updated_at"    TIMESTAMP NOT NULL DEFAULT(CURRENT_TIMESTAMP)
);
```

接下來可以執行 `npx prisma introspect` 來生成 IDL 如以下語法:

```
model Users {
  userId    String   @id @map("user_id")
  userName  String   @map("user_name")
  createdAt DateTime @default(now()) @map("created_at")
  updatedAt DateTime @default(now()) @map("updated_at")
  meetings  Meetings[]
}
```

使用IDL的目的是為了盡可能減少開發者自行定義 interface 如

```
// user.ts
interface User {
    userId: string;
    userName: string;
    createdAt: DateTime;
    updatedAt: DateTime;
    meetings: Meetings[]
}
```

盡可能由IDL來抽象出 TypeScript define file, 已經定義好輔助的指令

```
$ npm run map # 生成IDL 以及 TypeScript define file
$ npm run gen # 使用IDL 生成 TypeScript define file
```

倘若發現 IDL 定義的名稱不符合團隊開發風格, 可調整後僅使用 `npm run gen` 來生成定義最後於程式碼中使用

```
const { PrismaClient } = require('@prisma/client');
const { users } = new PrismaClient();
user.create(/* ... */);
user.findMany(/* ... */);
user.update(/* ... */);
```

開發階段, 可使用 `node ./prisma/seed.js` 來生成測試資料使用

Server

index.ts 與 application.ts

`index.ts` 為主要的伺服器啟用端口程式碼, 提供了 http 與 https 的伺服器, 開發者可根據產品方向選用適合的服務, `application.ts` 則是在伺服器封裝成 http 與 https 之前,應該處理得好的伺服器路由

shared

`interface.ts` 提供了主要的資料訪問介面, 來提開應用開發的嚴謹性, `gapi.ts` 則是封裝好Google OAuth API 的常用功能

routes

裡面提供 google, meeting, user 三大路由,

- `google.router` 當使用者授權Google給我們的應用後, 應該如何處理本應用與Google互動的方式
- `meeting.router` 新增、刪除、修改會議資訊
- `user.router` 使用者資料的查詢

Supplements

可以調整 `tsconfig.build` 與 `tsconfig` , 根據開發團隊的喜好, 也提供 `api.md` 文件供開發者查看更詳細的路由定義