```ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import * as compression from 'compression';
import helmet from 'helmet';
import { CorsOptions } from '@nestjs/common/interfaces/external/cors-options.interface';
import { SwaggerModule } from '@nestjs/swagger';
import { join } from 'path';
import { RequestMethod } from '@nestjs/common';
import { ExpressAdapter, NestExpressApplication } from '@nestjs/platform-express';
import * as hbs from 'hbs';
import * as cookieParser from 'cookie-parser';
import * as express from 'express';
import * as http from 'http';
import * as https from 'https';
import { httpsOptions, swaggerOptions, TModeApplication } from './app.config';


async function bootstrap() {
  const server = express();

  const app = await NestFactory.create<NestExpressApplication>(AppModule, new ExpressAdapter(server));

  // set template engine
  app.setBaseViewsDir(join(__dirname, '..', 'views'));
  hbs.registerHelper('eq', (a, b) => a === b);
  hbs.registerPartials(join(__dirname, '..', 'views/partials'));
  hbs.registerPartials(join(__dirname, '..', 'views/layout'));
  app.setViewEngine('hbs');

  // set global prefix
  app.setGlobalPrefix('api', {
    exclude: [{ path: 'admin/(.*)', method: RequestMethod.ALL }],
  });

  // cookie
  app.use(cookieParser());

  // compression
  app.use(compression());

  // CORS
  const corsOptions: CorsOptions = {
    origin: '*',
    methods: 'GET,HEAD,PUT,POST,DELETE',
  };
  app.enableCors(corsOptions);

  // Set HELMET
  app.use(helmet());


  const modeApplication: TModeApplication = (process.env.APP_ENV as TModeApplication) || 'local';

  if (modeApplication === 'production') {
    await app.init();

    https.createServer(httpsOptions, server).listen(8112, () => {
```

```
      console.log(`Server running on ${modeApplication} mode`)
    })
  } else if (modeApplication === 'development') {
    const document = SwaggerModule.createDocument(app, swaggerOptions);
    SwaggerModule.setup('doc', app, document);

    await app.init();

    https.createServer(httpsOptions, server).listen(8112, () => {
      console.log(`Server running on ${modeApplication} mode`)
    })
  } else {
    const document = SwaggerModule.createDocument(app, swaggerOptions);
    SwaggerModule.setup('doc', app, document);

    await app.init();

    http.createServer(server).listen(8112, () => {
      console.log(`Server running on ${modeApplication} mode`)
    })
  }
}

bootstrap();

```