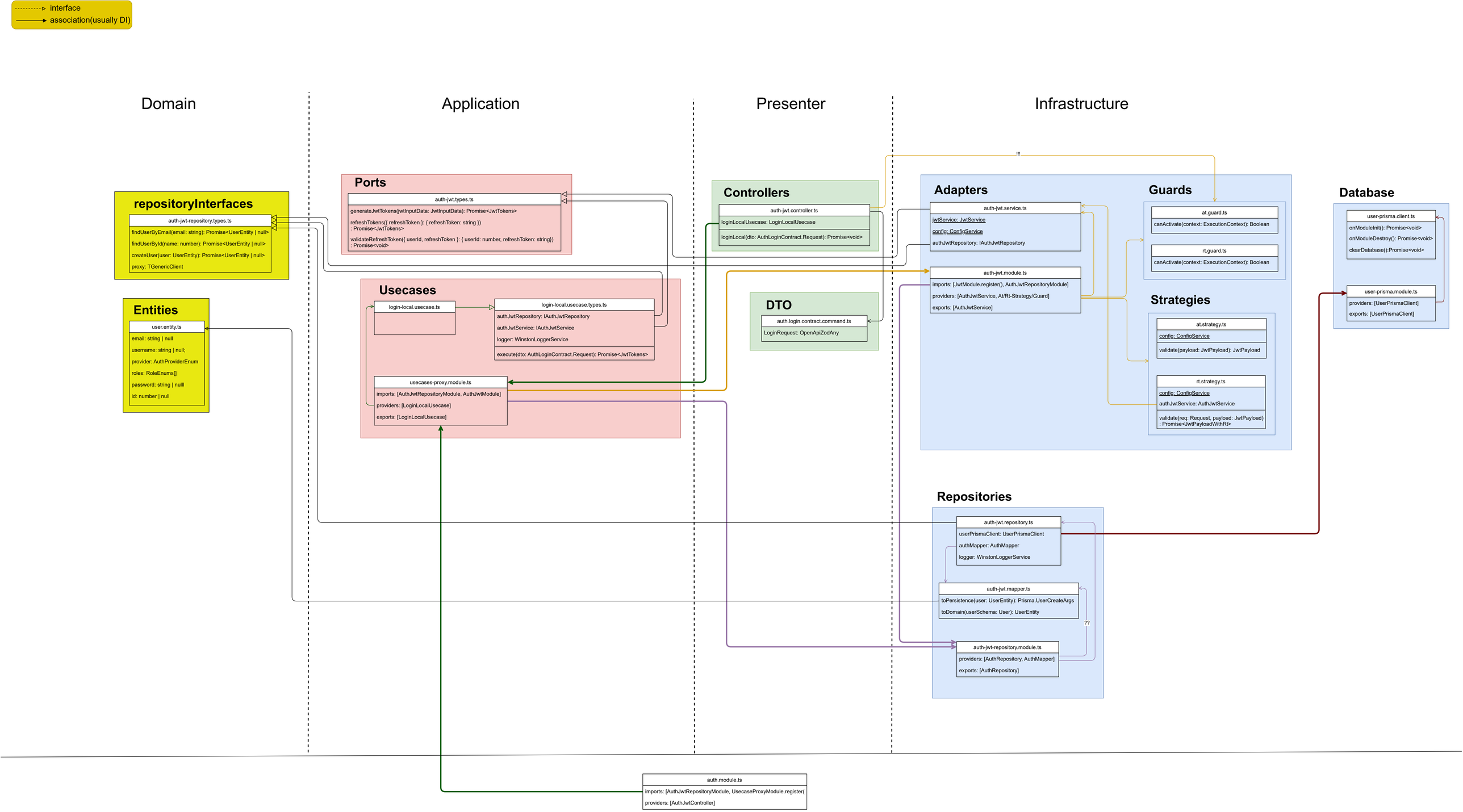
interface
association(usually DI)
Domain
Application
Presenter
Infrastructure
repositoryInterfaces
auth-jwt-repository.types.ts
findUserByEmail(email: string): Promise<UserEntity | null>
findUserById(name: number): Promise<UserEntity | null>
createUser(user: UserEntity): Promise<UserEntity | null>
proxy: TGenericClient
Entities
user.entity.ts
email: string | null
username: string | null;
provider: AuthProviderEnum
roles: RoleEnums[]
password: string | nulll
id: number | null
Ports
auth-jwt.types.ts
generateJwtTokens(jwtInputData: JwtInputData): Promise<JwtTokens>
refreshTokens({ refreshToken }: { refreshToken: string })
: Promise<JwtTokens>
validateRefreshToken({ userId, refreshToken }: { userId: number, refreshToken: string})
: Promise<void>
Usecases
login-local.usecase.ts
login-local.usecase.types.ts
authJwtRepository: IAuthJwtRepository
authJwtService: IAuthJwtService
logger: WinstonLoggerService
execute(dto: AuthLoginContract.Request): Promise<JwtTokens>
usecases-proxy.module.ts
imports: [AuthJwtRepositoryModule, AuthJwtModule]
providers: [LoginLocalUsecase]
exports: [LoginLocalUsecase]
Controllers
auth-jwt.controller.ts
loginLocalUsecase: LoginLocalUsecase
loginLocal(dto: AuthLoginContract.Request): Promise<void>
DTO
auth.login.contract.command.ts
LoginRequest: OpenApiZodAny
!!!
Adapters
auth-jwt.service.ts
jwtService: JwtService
config: ConfigService
authJwtRepository: IAuthJwtRepository
auth-jwt.module.ts
imports: [JwtModule.register(), AuthJwtRepositoryModule]
providers: [AuthJwtService, At/Rt-Strategy/Guard]
exports: [AuthJwtService]
Guards
at.guard.ts
canActivate(context: ExecutionContext): Boolean
rt.guard.ts
canActivate(context: ExecutionContext): Boolean
Strategies
at.strategy.ts
config: ConfigService
validate(payload: JwtPayload): JwtPayload
rt.strategy.ts
config: ConfigService
authJwtService: AuthJwtService
validate(req: Request, payload: JwtPayload)
: Promise<JwtPayloadWithRt>
Database
user-prisma.client.ts
onModuleInit(): Promise<void>
onModuleDestroy(): Promise<void>
clearDatabase():Promise<void>
user-prisma.module.ts
providers: [UserPrismaClient]
exports: [UserPrismaClient]
Repositories
auth-jwt.repository.ts
userPrismaClient: UserPrismaClient
authMapper: AuthMapper
logger: WinstonLoggerService
auth-jwt.mapper.ts
toPersistence(user: UserEntity): Prisma.UserCreateArgs
toDomain(userSchema: User): UserEntity
??
auth-jwt-repository.module.ts
providers: [AuthRepository, AuthMapper]
exports: [AuthRepository]
auth.module.ts
imports: [AuthJwtRepositoryModule, UsecaseProxyModule.register(
providers: [AuthJwtController]