Domain Layer

User <<Pattern: Entity>>
-id: UserId
-email: Email
-passwordHash: string
-role: UserRole
-hasPremium: boolean
+upgradeToPremium(): void
+verifyEmail(): void
+upgradeToPremium(): void

Class <<Pattern: Entity>>
-id: ClassId
-code: ClassCode
-teacherId: UserId
-students: UserId[]
+addStudent(userId): void
+removeStudent(userId): void
+generateNewCode(): ClassCode

DomainException
+message: string
+code: string
InvalidEmailException
UserNotFoundException
ClassFullException

Email <<ValueObject>>
-value: string
-email: Email
-passwordHash: string
-role: UserRole
-hasPremium: boolean
+{static} create(email): Email
+getValue(): string
Pattern: Value Object

<<enumeration>> UserRole
PADRE
DOCENTE
ADMIN

ClassCode
-value: string
+code: string
+{static} generate(): ClassCode
+{static} create(code): ClassCode
+getValue(): string

NestJS Module System

AppModule
imports: [AuthModule, ContentModule, ClassModule]
Pattern: Dependency Injection

AuthModule
providers:
LoginUseCase
UserRepositoryImpl
BcryptHashService
JwtStrategy
exports: [JwtStrategy]

ContentModule
Providers
GetLessonsUseCase
GetLessonsUseCase
CacheService

Presentation Layer

ClassController
+createClass(dto): Promise<ClassDto>
+getMyClasses(): Promise<ClassDto[]>
+joinClass(code): Promise<void>
+getStudents(classId): Promise<StudentDto[]>

ContentController
+getLevels(): Promise<LevelDto[]>
+getModules(levelId): Promise<ModuleDto[]>
+getLessons(moduleId): Promise<LessonDto[]>
+getActivities(lessonId): Promise<ActivityDto[]>

AuthController
+login(dto): Promise<TokenDto>
+register(dto): Promise<UserDto>
+verifyEmail(token): Promise<void>
+refreshToken(): Promise<TokenDto>
Decorators:
@UseGuards(JwtAuthGuard)
@UseInterceptors(LoggingInterceptor)

JwtAuthGuard
+canActivate(context): boolean
-jwtStrategy: JwtStrategy
Pattern: Strategy

TransformInterceptor
+intercept(context, next): Observable
Pattern: Decorator

LoggingInterceptor
+intercept(context, next): Observable
Pattern: Decorator

Application Layer

UserMapper
+toDto(entity): UserDto
+toEntity(dto): User
Pattern: Mapper/DTO

CreateClassUseCase
-classRepository: IClassRepository
-codeGenerator: ICodeGenerator
+execute(dto): ClassDto

LoginUseCase
-userRepository: IUserRepository
-jwtService: JwtService
-hashService: IHashService
+execute(email, password): TokenDto
Pattern: Command

IHashService
+hash(password): string
+compare(plain, hash): boolean

GetLessonsUseCase
-contentRepository: IContentRepository
+execute(moduleId): LessonDto[]

RegisterUseCase
-userRepository: IUserRepository
-hashService: IHashService
-emailService: IEmailService
+execute(dto): UserDto

IUserRepository
+findByEmail(email): User
+save(user): User
+findById(id): User

IContentRepository
+getLevels(): Level[]
+getModules(levelId): Module[]
+getLessons(moduleId): Lesson[]

<<permit>>

Infraestructure Layer

BcryptHashService
+hash(password): string
+compare(plain, hash): boolean
Pattern: Adapter

UserRepositoryImpl
-prisma: PrismaService
+findByEmail(email): User
+save(user): User
+findById(id): User
Pattern: Repository

ContentRepositoryImpl
-prisma: PrismaService
-cacheService: CacheService
+getLevels(): Level[]
+getModules(levelId): Module[]

PrismaService
-client: PrismaClient
+onModuleInit(): void
+onModuleDestroy(): void
Pattern: Singleton

CacheService
-cache: Map<string, any>
+get<T>(key): T
+set(key, value, ttl): void
+invalidate(pattern): void

JwtStrategy
+validate(payload): User
Pattern: Strategy

Comunicación HTTP/REST

HTTP Response
status: 200 | 401 | 404 | 500
body: JSON
{data: {...}, message "..."}

HTTP Request
POST /auth/login
GET /content/levels
POST /classes
Headers: Authorization: Bearer <token>
body: JSON

Pattern: DTO (Data Transfer Object)

App Flutter
Frontend

Dependency Injection

injection_container
+init(): Future<void>
Pattern: Composition Root

Registra todas las dependencias al inicio

GetIt (sl)
+instance: GetIt
+registerSingleton<T>(instance)
+registerFactory<T>(factory)
+registerLazySingleton<T>(factory)
+call<T>(): T
Pattern: Service Locator

Data Layer

AuthRepositoryImpl
-remoteDataSource: AuthRemoteDataSource
-localDataSource: AuthLocalDataSource
-networkInfo: NetworkInfo
+login(email, password): Either<Failure, User>
Pattern: Repository

ContentRepositoryImpl
-remoteDataSource: ContentRemoteDataSource
-localDataSource: ContentLocalDataSource
-networkInfo: NetworkInfo
+getLevels(): Either<Failure, List<Level>>
Offline-first strategy

AuthRemoteDataSource
-apiClient: ApiClient
+login(email, password): UserModel
+register(dto): UserModel
Pattern: Adapter

AuthLocalDataSource
-secureStorage: FlutterSecureStorage
-prefs: SharedPreferences
+cacheToken(token): void
+getToken(): String?
+cacheUser(user): void

NetworkInfo
-connectivity: Connectivity
+isConnected: Future<bool>
Verifica conectividad

ContentLocalDataSource
-cacheService: CacheService
+cacheLevels(levels): void
+getCachedLevels(): List<Level>?
Offline Support

ApiClient
-dio: Dio
-secureStorage: FlutterSecureStorage
+get(path): Response
+post(path, data): Response
-_addAuthHeader(): void
Pattern: Singleton + Facade

UserModel
+id: String
+email: String
+name: String
+role: String
+fromJson(json): UserModel
+toJson(): Map
+toEntity(): User
Pattern: DTO/Model

Core/widgets

MainHomeWrapper
+build(context): Widget
Pattern: Strategy
-_buildLevels(): Widget

BlocProvider
+create: BLoC
+child: Widget
Pattern: Factory

BlocBuilder
+bloc: BLoC
+builder: (context, state) => Widget
Pattern: Observer

Data Layer

SchoolClass
+id: String
+name: String
+code: String
+studentsCount: int

LoginUseCase
-repository: AuthRepository
+call(email, password): Either<Failure, User>
Pattern: Command

GetLessonsUseCase
-repository: ContentRepository
+call(moduleId): Either<Failure, List<Lesson>>
Pattern: Command

User
+id: String
+email: String
+name: String
+role: UserRole
+isPadre: bool
+isDocente: bool

AuthRepository
+login(email, password): Either<Failure, User>
+register(dto): Either<Failure, User>
+logout(): Either<Failure, void>
+getCurrentUser(): Either<Failure, User>

ContentRepository
+getLevels(): Either<Failure, List<Level>>
+getModules(levelId): Either<Failure, List<Module>>
+getLessons(moduleId): Either<Failure, List<Lesson>>

User
+id: String
+name: String
+level: LevelType
+isUnlocked: bool

Failure
ServerFailure
CacheFailure
NetworkFailure
Pattern: Either/Result

Presentation Layer

HomePage
+build(context): Widget
-_buildHeader(): Widget
-_buildLevels(): Widget

TeacherHomePage
+build(context): Widget
-_buildClassesList(): Widget
-_buildStats(): Widget

LoginPage
+build(context): Widget
-_formKey: GlobalKey
-_onLogin(): void

ContentBloc
-getLessonsUseCase: GetLessonsUseCase
-getModulesUseCase: GetModulesUseCase
+Stream<ContentState> mapEventToState
Pattern: Command

ClassBloc
-createClassUseCase: CreateClassUseCase
-getTeacherClassesUseCase: GetTeacherClassesUseCase
+Stream<ClassState> mapEventToState
Pattern: Command

AuthBloc
-loginUseCase: LoginUseCase
-logoutUseCase: LogoutUseCase
+Stream<AuthState> mapEventToState
Pattern: Observer/BLoC

AuthEvent
LoginEvent
LogoutEvent
CheckStatusEvent
RegisterEvent

AuthState
AuthInitialState
AuthLoadingState
AuthAuthenticatedState
AuthUnauthenticatedState
AuthErrorState