NotificacionesNotifier

+ obtenerListadoNotificaciones(page: int): void

VerNotificacionesScreen

NotificacionesState

- notificaciones: List<NotificacionEntity>

+ copyWith()

NotificacionEntity

- id: int

- titulo: String

- descripcion: String

- linkInterno: String

- leido: bool

0..n

ObtenerNotificacionesUseCase

+ execute(page: int?): Future<List<NotificacionEntity>>

MarcarLeidaUseCase

+ execute(idNotificacion: int?): Future<bool>

<<Interface>>
NotificacionesRepository

+ obtenerListadoNotificaciones(page: int): List<NotificacionEntity>

+ marcarComoLeida(idNotificaciones: List<int>): Future<bool>

NotificacionesLocalModel

+ data: List<Map<String, dynamic>>
+ perPage: int
+ total: int
+ lastPage: int
+ currentPage: int

+ toEntityList(): List<NotificacionEntity>

NotificacionesRepositoryImpl

+ obtenerListadoNotificaciones(page: int): List<NotificacionEntity>

+ marcarComoLeida(idNotificaciones: List<int>): Future<bool>

NotificacionesLocalDataSource

+ obtenerListadoNotificaciones(page: int): Future<NotificacionesLocalModel>

+ marcarComoLeida(idNotificacion: int): Future<bool>

<>
NotificacionesResponseModel

+ statusCode: int

NotificacionesRemoteDataSource

+ obtenerListadoNotificaciones(page: int): Future<NotificacionesResponseModel>

+ marcarComoLeida(idNotificacion: int): Future<bool>

Extends

NotificacionesErrorResponse

+ statusCode: int

+ message: String

Extends

```
+ message: String
```
```
+ fromJson(statusCode: int, json: Map<String, dynamic>): NotificacionesErrorResponse

+ toJson():  Map<String, dynamic>
```

NotificacionesSuccessResponse

```
+ statusCode: int
```
```
+ fromJson(statusCode: int, json: Map<String, dynamic>): NotificacionesSuccessResponse

+ toEntityList():  List<NotificacionEntity>
```

<<mixin>>
PaginatedResponseMixin
```
+ data: List<Map<String, dynamic>>
+ currentPage: int
+ lastPage: int
+ total: int
+ perPage: int
```

─Extends─

DatosPagoModel
- fechaPago: DateTime
- importeTotal: double
- metodoPago: String
- titularTarjeta: String?
- ultimos4DigitosTarjeta: String

ComprobanteEntity
- idOperacion: int
- caratula: String
- boletasPagadas: List<BoletaEntity>

PaginatedResponseModel
- statusCode: int
- data: List<Map<String, dynamic>>
- currentPage: int
- lastPage: int
- total: int
- perPage: int
+ fromJson(json: Map<String, dynamic>): PaginatedResponseModel

ComprobanteState
- comprobante: ComprobanteEntity
+ copyWith()

ObtenerComprobanteUseCase
+ execute(idBoletaPagada: int): Future<BoletaEntity>

DescargarComprobanteUseCase
+ execute(caratula: String, monto: double?): Future<BoletaEntity>

ComprobanteNotifier
+ obtenerComprobante(idBoletaPagada: int): void
+ descargarComprobante(): void
+ crearBoletaFin(boletaFinData: BoletaFinDataState): Future<BoletaEntity>
+ buscarBoletasInicioPagadas(page: int?, caratulaBuscada: String?): Future<void>
+ irAPagina(page: int): Future<void>
+ irAPaginaSiguiente(): Future<void>
+ irAPaginaAnterior(): Future<void>
+ refresh(): Future<void>
- convertirBoletaHistorialAEntity(model: BoletaHistorialModel): BoletaEntity
- convertirBoletaInicioPagadaAEntity(model: BoletaInicioPagadaModel): BoletaEntity
- getFromLocalCache(page: int): Future<BoletasState>
- syncCache(): Future<void>
- handleConnectivityChange(previous: ConnectivityStatus?, current: ConnectivityStatus): void

<<Interface>>
BoletasRepository
+ crearBoletaInicio(caratula: String, nroAfiliado: int, monto: double): Future<BoletaEntity>
+ crearBoletaFinalizacion(nroAfiliado, caratula, idBoletaInicio, monto, fechaRegulacion, honorarios, cantidadJus, valorJus, nroExpediente, anioExpediente, cuij): Future<BoletaEntity>
+ obtenerHistorialBoletas(nroAfiliado: int, page: int?): Future<HistorialBoletasSuccessResponse>
+ buscarBoletasInicioPagadas(nroAfiliado: int, page: int?, caratulaBuscada: String?): Future<PaginatedResponseModel>
+ obtenerComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>
+ descargarComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>

<<enumeration>>
ConnectivityStatus
online
offline

ConnectivityNotifier
- connectivitySubscription: StreamSubscription<List<ConnectivityResult>>
- checkInternetConnection(): Future<bool>

BoletasRepositoryImpl
+ crearBoletaInicio(caratula: String, nroAfiliado: int, monto: double): Future<BoletaEntity>
+ crearBoletaFinalizacion(nroAfiliado, caratula, idBoletaInicio, monto, fechaRegulacion, honorarios, cantidadJus, valorJus, nroExpediente, anioExpediente, cuij): Future<BoletaEntity>
+ obtenerHistorialBoletas(nroAfiliado: int, page: int?): Future<HistorialBoletasSuccessResponse>
+ buscarBoletasInicioPagadas(nroAfiliado: int, page: int?, caratulaBuscada: String?): Future<PaginatedResponseModel>
+ obtenerComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>
+ descargarComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>

VerComprobanteScreen

HistorialBoletasScreen

MainNavigationScreen

BoletasLocalDataSource
- client: HttpClient
+ descargarComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>
+ obtenerBoletasLocales(limit: int?, offset: int?, caratulaFiltro: String?): Future<List<BoletaEntity>>
+ obtenerConteoBoletasLocales(caratulaFiltro: String?): Future<int>
+ limpiarCache(): void
+ obtenerUltimaSincronizacion(): Future<Datetime?>
+ tieneBoletasEnCache(): Future<bool>
+ boletaToMap(boleta: BoletaEntity): Map<String, dynamic>
+ mapToBoleta(map: Map<String, dynamic>): BoletaEntity
+ stringToBoletaTipo(tipo: String): BoletaTipo
+ guardarBoletas(boletas: List<BoletaEntity>): void
+ obtenerComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>

NavigationNotifier
- state: int
- selectTab(index: int): void

BoletasDataSource
- client: HttpClient
+ crearBoletaInicio(caratula: String, dato1: String): Future<CrearBoletaResponse>
+ crearBoletaFinalizacion(id_boleta_inicio: int, dato1: String): Future<CrearBoletaResponse>
+ obtenerHistorialBoletas(nroAfiliado: int, page: int?, mostrarPagadas: int): Future<HistorialBoletasResponse>
+ buscarBoletasInicioPagadas(nroAfiliado: int, page: int?, caratulaBuscada: String?): Future<PaginatedResponseModel>
+ obtenerComprobante(idBoletaPagada: int): Future<ComprobanteDataModel>