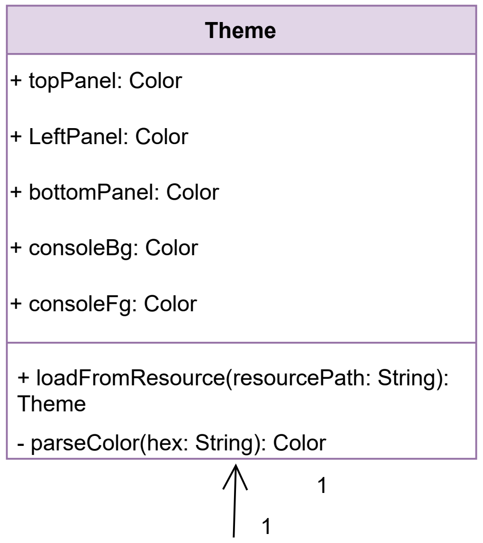
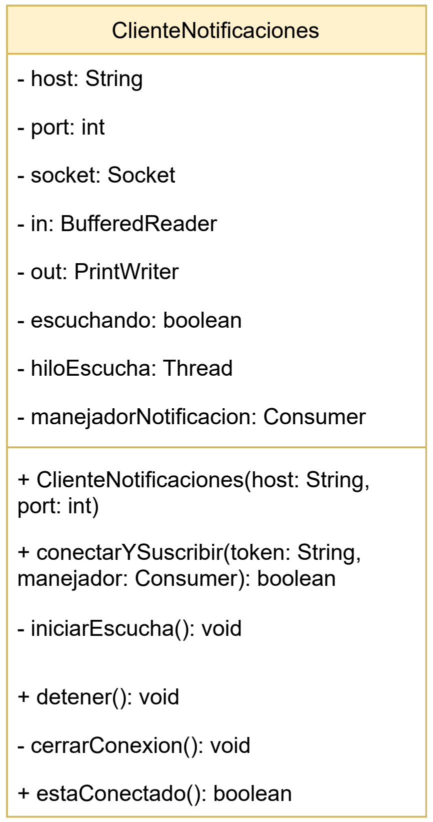

Theme
+ topPanel: Color
+ LeftPanel: Color
+ bottomPanel: Color
+ consoleBg: Color
+ consoleFg: Color
+ loadFromResource(resourcePath: String): Theme
- parseColor(hex: String): Color
1
1



ClienteNotificaciones

- host: String
- port: int
- socket: Socket
- in: BufferedReader
- out: PrintWriter
- escuchando: boolean
- hiloEscucha: Thread
- manejadorNotificacion: Consumer

+ ClienteNotificaciones(host: String, port: int)

+ conectarYSuscribir(token: String, manejador: Consumer): boolean

- iniciarEscucha(): void

+ detener(): void

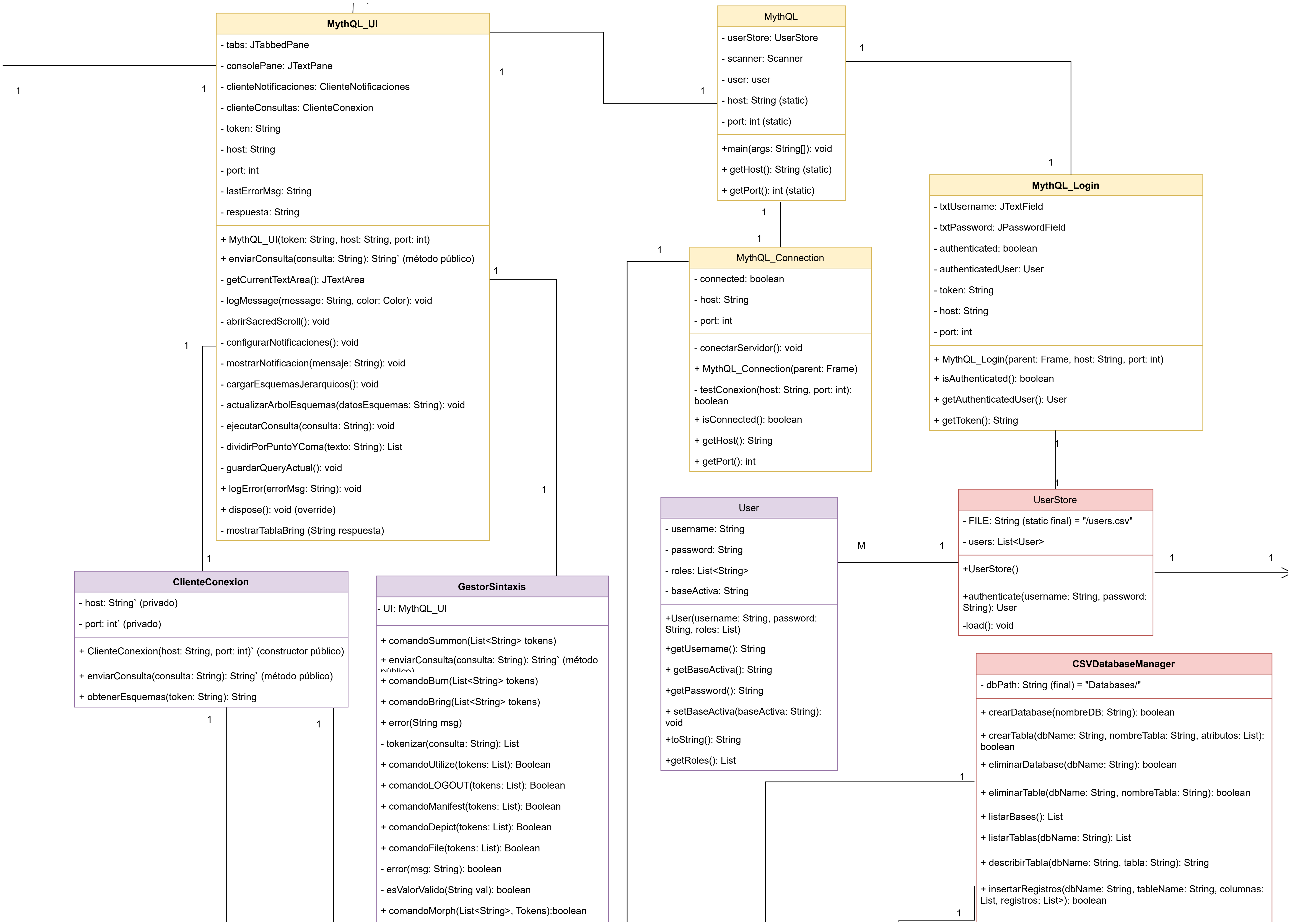
- cerrarConexion(): void

+ estaConectado(): boolean

CLIENTE

LOGICA

SERVIDOR



MythQL_UI

- tabs: JTabbedPane
- consolePane: JTextPane
- clienteNotificaciones: ClienteNotificaciones
- clienteConsultas: ClienteConexion
- token: String
- host: String
- port: int
- lastErrorMsg: String
- respuesta: String

+ MythQL_UI(token: String, host: String, port: int)
+ enviarConsulta(consulta: String): String` (método público)
- getCurrentTextArea(): JTextArea
- logMessage(message: String, color: Color): void
- abrirSacredScroll(): void
- configurarNotificaciones(): void
- mostrarNotificacion(mensaje: String): void
- cargarEsquemasJerarquicos(): void
- actualizarArbolEsquemas(datosEsquemas: String): void
- ejecutarConsulta(consulta: String): void
- dividirPorPuntoYComa(texto: String): List
- guardarQueryActual(): void
+ logError(errorMsg: String): void
+ dispose(): void (override)
- mostrarTablaBring (String respuesta)

MythQL

- userStore: UserStore
- scanner: Scanner
- user: user
- host: String (static)
- port: int (static)

+main(args: String[]): void
+ getHost(): String (static)
+ getPort(): int (static)

MythQL_Connection

- connected: boolean
- host: String
- port: int

- conectarServidor(): void
+ MythQL_Connection(parent: Frame)
- testConexion(host: String, port: int): boolean
+ isConnected(): boolean
+ getHost(): String
+ getPort(): int

MythQL_Login

- txtUsername: JTextField
- txtPassword: JPasswordField
- authenticated: boolean
- authenticatedUser: User
- token: String
- host: String
- port: int

+ MythQL_Login(parent: Frame, host: String, port: int)
+ isAuthenticated(): boolean
+ getAuthenticatedUser(): User
+ getToken(): String

User

- username: String
- password: String
- roles: List<String>
- baseActiva: String

+User(username: String, password: String, roles: List)
+getUsername(): String
+ getBaseActiva(): String
+getPassword(): String
+ setBaseActiva(baseActiva: String): void
+toString(): String
+getRoles(): List

UserStore

- FILE: String (static final) = "/users.csv"
- users: List<User>

+UserStore()
+authenticate(username: String, password: String): User
-load(): void

ClienteConexion

- host: String` (privado)
- port: int` (privado)

+ ClienteConexion(host: String, port: int)` (constructor público)
+ enviarConsulta(consulta: String): String` (método público)
+ obtenerEsquemas(token: String): String

GestorSintaxis

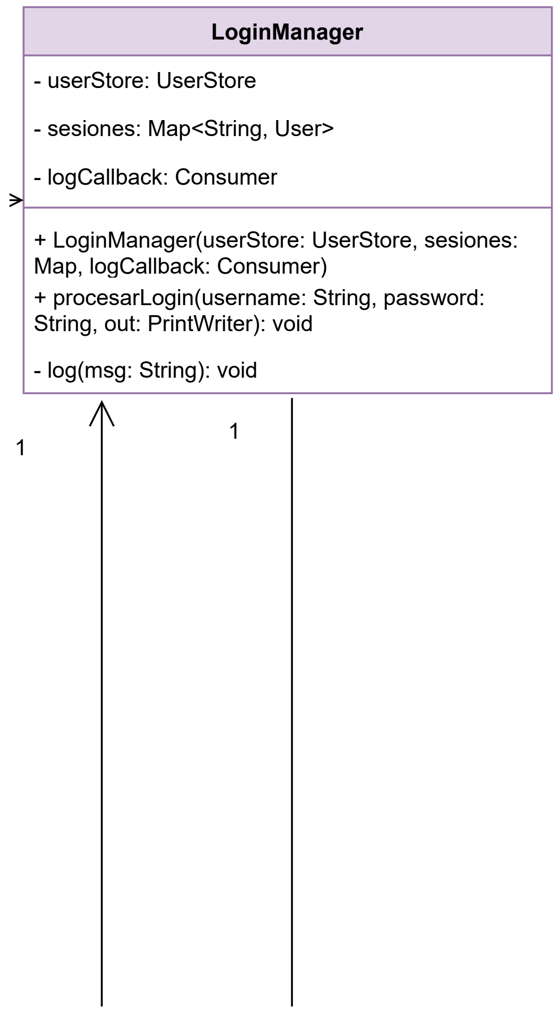
- UI: MythQL_UI

+ comandoSummon(List<String> tokens)
+ enviarConsulta(consulta: String): String` (método público)
+ comandoBurn(List<String> tokens)
+ comandoBring(List<String> tokens)
+ error(String msg)
- tokenizar(consulta: String): List
+ comandoUtilize(tokens: List): Boolean
+ comandoLOGOUT(tokens: List): Boolean
+ comandoManifest(tokens: List): Boolean
+ comandoDepict(tokens: List): Boolean
+ comandoFile(tokens: List): Boolean
- error(msg: String): boolean
- esValorValido(String val): boolean
+ comandoMorph(List<String>, Tokens):boolean

CSVDatabaseManager

- dbPath: String (final) = "Databases/"

+ crearDatabase(nombreDB: String): boolean
+ crearTabla(dbName: String, nombreTabla: String, atributos: List): boolean
+ eliminarDatabase(dbName: String): boolean
+ eliminarTable(dbName: String, nombreTabla: String): boolean
+ listarBases(): List
+ listarTablas(dbName: String): List
+ describirTabla(dbName: String, tabla: String): String
+ insertarRegistros(dbName: String, tableName: String, columnas: List, registros: List>): boolean



LoginManager

- userStore: UserStore

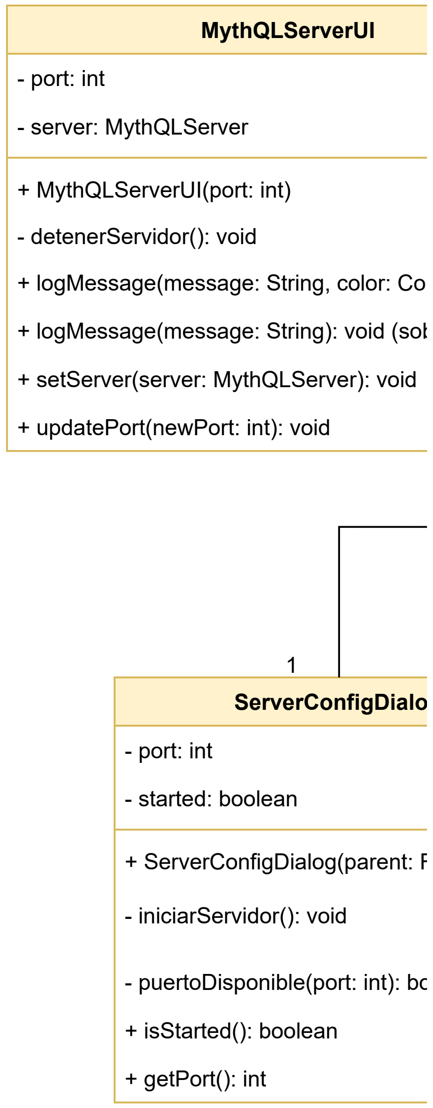
- sesiones: Map<String, User>

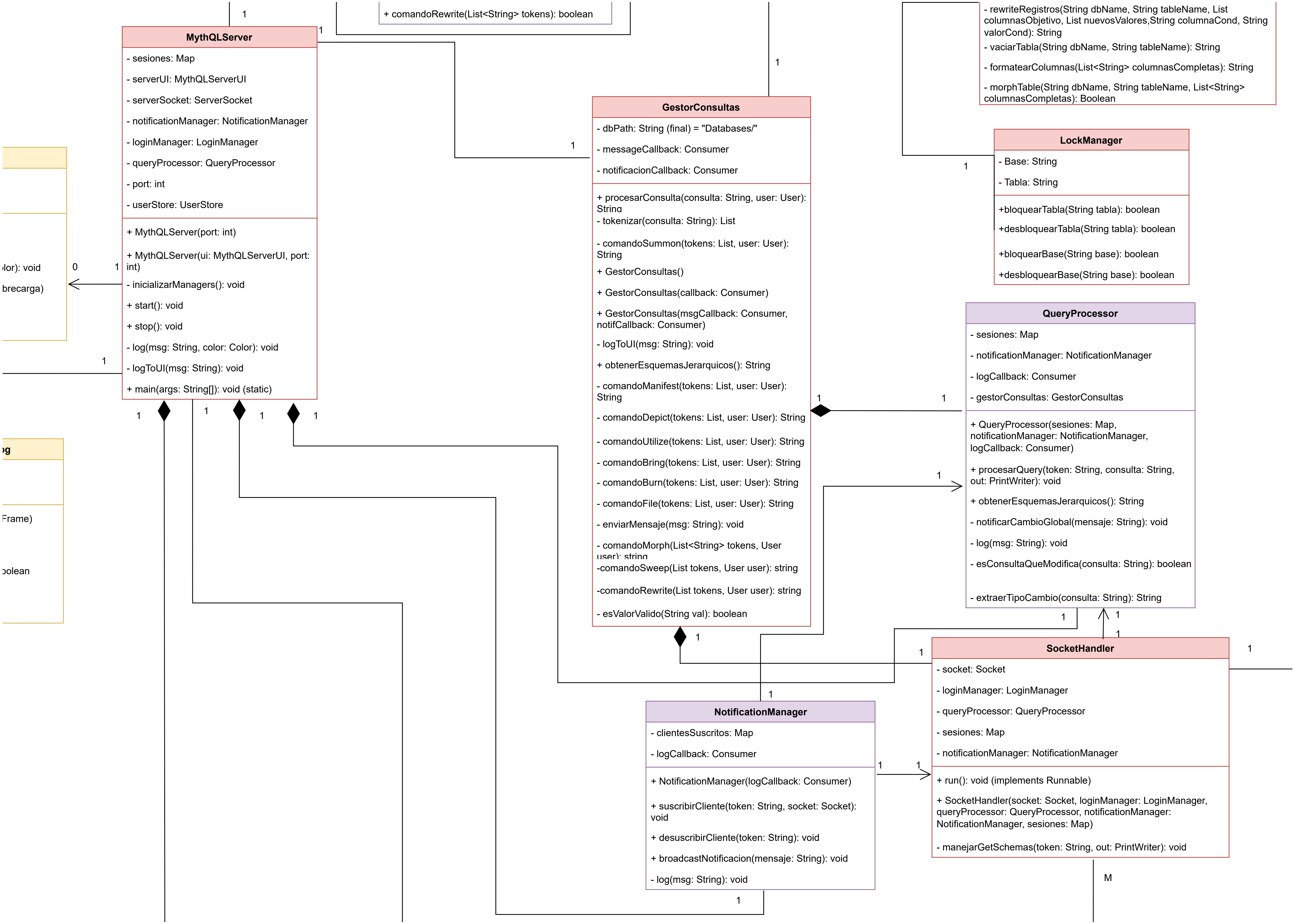
- logCallback: Consumer

+ LoginManager(userStore: UserStore, sesiones: Map, logCallback: Consumer)

+ procesarLogin(username: String, password: String, out: PrintWriter): void

- log(msg: String): void


MythQLServerUI
- port: int
- server: MythQLServer
+ MythQLServerUI(port: int)
- detenerServidor(): void
+ logMessage(message: String, color: Co
+ logMessage(message: String): void (sob
+ setServer(server: MythQLServer): void
+ updatePort(newPort: int): void
1
ServerConfigDialo
- port: int
- started: boolean
+ ServerConfigDialog(parent: F
- iniciarServidor(): void
- puertoDisponible(port: int): bo
+ isStarted(): boolean
+ getPort(): int



+ comandoRewrite(List<String> tokens): boolean

- rewriteRegistros(String dbName, String tableName, List columnasObjetivo, List nuevosValores,String columnaCond, String valorCond): String
- vaciarTabla(String dbName, String tableName): String
- formatearColumnas(List<String> columnasCompletas): String
- morphTable(String dbName, String tableName, List<String> columnasCompletas): Boolean

MythQLServer

- sesiones: Map
- serverUI: MythQLServerUI
- serverSocket: ServerSocket
- notificationManager: NotificationManager
- loginManager: LoginManager
- queryProcessor: QueryProcessor
- port: int
- userStore: UserStore

+ MythQLServer(port: int)
+ MythQLServer(ui: MythQLServerUI, port: int)
- inicializarManagers(): void
+ start(): void
+ stop(): void
- log(msg: String, color: Color): void
- logToUI(msg: String): void
+ main(args: String[]): void (static)

lor): void
brecarga)

og

Frame)

oolean

GestorConsultas

- dbPath: String (final) = "Databases/"
- messageCallback: Consumer
- notificacionCallback: Consumer

+ procesarConsulta(consulta: String, user: User): String
- tokenizar(consulta: String): List
- comandoSummon(tokens: List, user: User): String
+ GestorConsultas()
+ GestorConsultas(callback: Consumer)
+ GestorConsultas(msgCallback: Consumer, notifCallback: Consumer)
- logToUI(msg: String): void
+ obtenerEsquemasJerarquicos(): String
- comandoManifest(tokens: List, user: User): String
- comandoDepict(tokens: List, user: User): String
- comandoUtilize(tokens: List, user: User): String
- comandoBring(tokens: List, user: User): String
- comandoBurn(tokens: List, user: User): String
- comandoFile(tokens: List, user: User): String
- enviarMensaje(msg: String): void
- comandoMorph(List<String> tokens, User user): string
-comandoSweep(List tokens, User user): string
-comandoRewrite(List tokens, User user): string
- esValorValido(String val): boolean

LockManager

- Base: String
- Tabla: String

+bloquearTabla(String tabla): boolean
+desbloquearTabla(String tabla): boolean
+bloquearBase(String base): boolean
+desbloquearBase(String base): boolean

QueryProcessor

- sesiones: Map
- notificationManager: NotificationManager
- logCallback: Consumer
- gestorConsultas: GestorConsultas

+ QueryProcessor(sesiones: Map, notificationManager: NotificationManager, logCallback: Consumer)
+ procesarQuery(token: String, consulta: String, out: PrintWriter): void
+ obtenerEsquemasJerarquicos(): String
- notificarCambioGlobal(mensaje: String): void
- log(msg: String): void
- esConsultaQueModifica(consulta: String): boolean
- extraerTipoCambio(consulta: String): String

SocketHandler

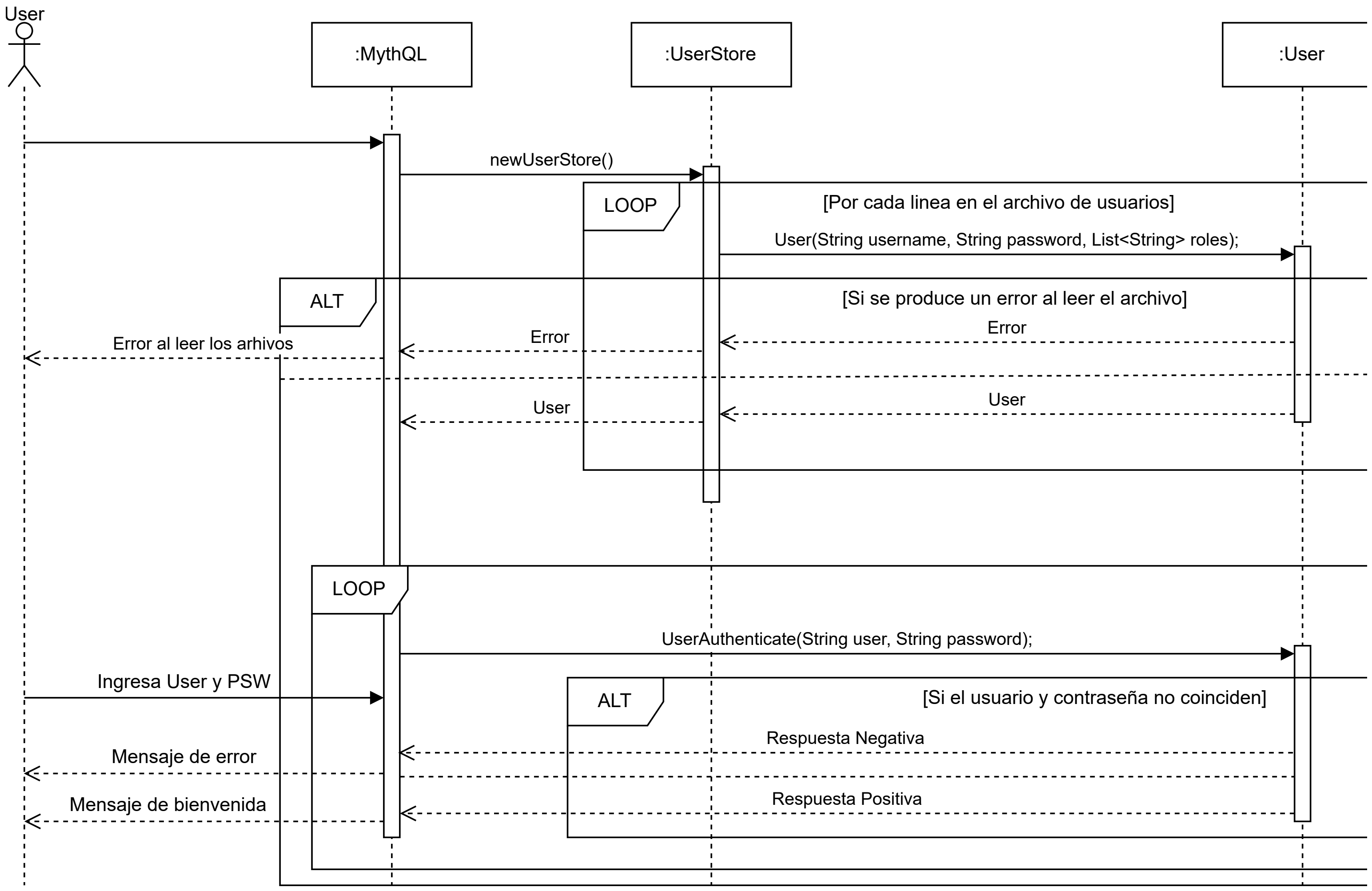
- socket: Socket
- loginManager: LoginManager
- queryProcessor: QueryProcessor
- sesiones: Map
- notificationManager: NotificationManager

+ run(): void (implements Runnable)
+ SocketHandler(socket: Socket, loginManager: LoginManager, queryProcessor: QueryProcessor, notificationManager: NotificationManager, sesiones: Map)
- manejarGetSchemas(token: String, out: PrintWriter): void

NotificationManager

- clientesSuscritos: Map
- logCallback: Consumer

+ NotificationManager(logCallback: Consumer)
+ suscribirCliente(token: String, socket: Socket): void
+ desuscribirCliente(token: String): void
+ broadcastNotificacion(mensaje: String): void
- log(msg: String): void


User
:MythQL
:UserStore
:User
newUserStore()
LOOP
[Por cada linea en el archivo de usuarios]
User(String username, String password, List<String> roles);
ALT
[Si se produce un error al leer el archivo]
Error
Error
Error al leer los arhivos
User
User
LOOP
UserAuthenticate(String user, String password);
Ingresa User y PSW
ALT
[Si el usuario y contraseña no coinciden]
Respuesta Negativa
Mensaje de error
Respuesta Positiva
Mensaje de bienvenida