Playlist Service — Complete Project Specification

> **Purpose**: This document contains all specifications needed for an IDE agent to implement the Playlist Service from scratch. It includes architecture, data models, API contracts, UI wireframes, and integration requirements.

TABLE OF CONTENTS

---

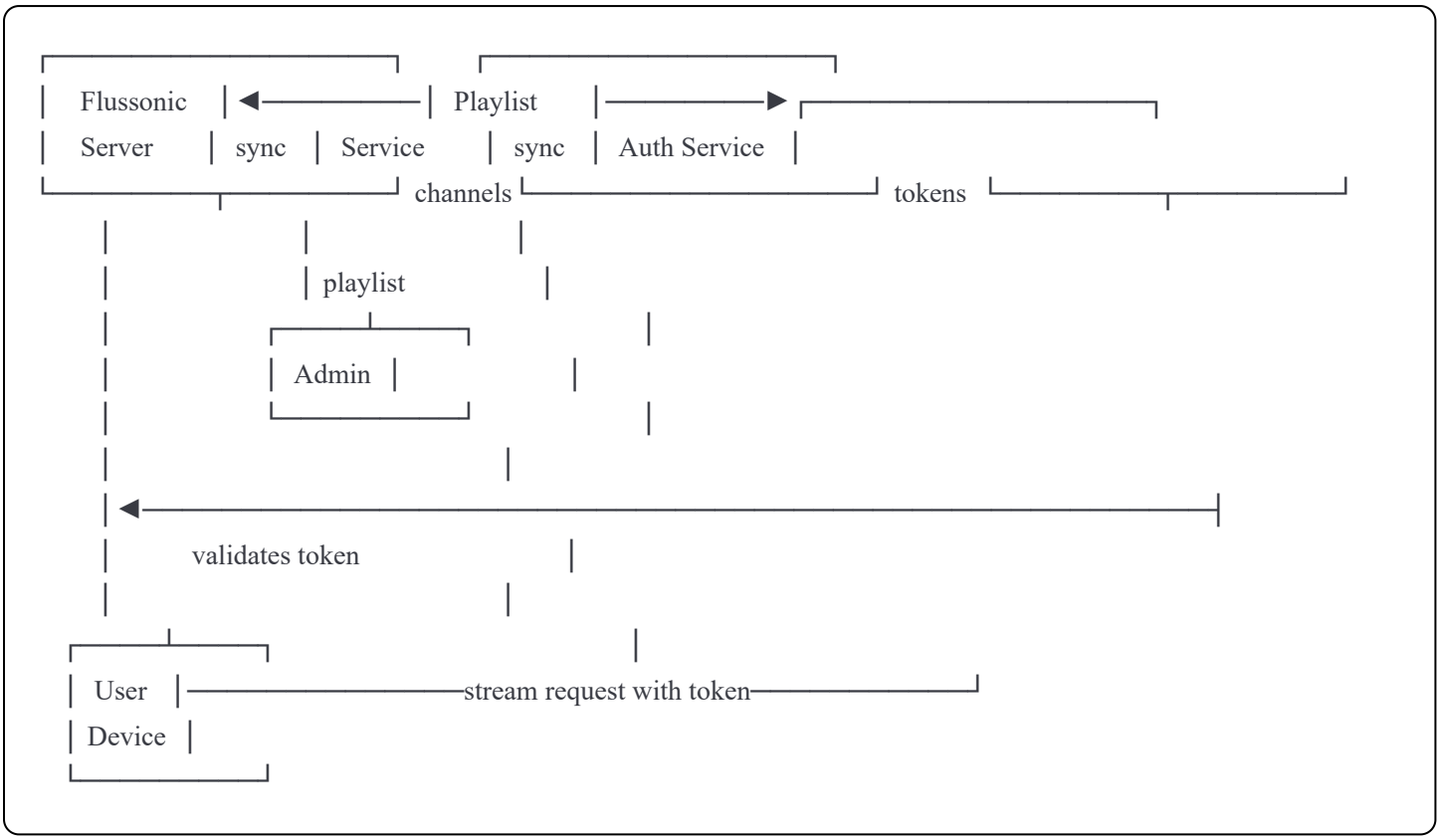
1. PROJECT OVERVIEW

1.1 Purpose

The Playlist Service is a middleware application that manages IPTV playlists for Flussonic Media Server. It allows administrators to:

- Sync channels from Flussonic (read-only cache)
- Organize channels into groups, packages, and tariffs
- Manage users with subscription-based channel access
- Generate personalized M3U/M3U8 playlists with authentication tokens
- Synchronize user tokens with an external Auth Service

1.2 System Context



1.3 Key Workflows

Channel Sync (Manual)

1. Admin triggers sync from UI
2. Service fetches channel list from Flussonic API
3. New channels are added, existing channels updated
4. Channels missing from Flussonic marked as "orphaned"
5. UI-managed fields (tvg_id, logo, group, sort_order) preserved

User Playlist Generation

1. Admin creates user with tariffs/packages/channels
2. Service generates unique token
3. Service registers token in Auth Service with allowed_streams
4. Admin downloads playlist file for user
5. Playlist contains all resolved channels with embedded token

Stream Authentication Flow

1. User device requests stream from Flussonic with token
2. Flussonic validates token via Auth Service
3. Auth Service checks token validity, session limits, allowed streams
4. Flussonic streams to user if authorized

---

2. TECHNOLOGY STACK

2.1 Backend

Component	Technology	Version	Notes
Language	Python	3.12+	Use modern type hints (T | None, list[T])
Framework	FastAPI	0.115+	Async, with lifespan context
ORM	SQLAlchemy	2.0+	Async with Mapped type hints
Validation	Pydantic	2.0+	Use model_config, ConfigDict
Server	Uvicorn	latest	ASGI server
Package Manager	uv	latest	Fast Python package manager
Database	PostgreSQL	15+	Primary database
DB Driver	asyncpg	latest	Async PostgreSQL driver
Migrations	Alembic	latest	Database migrations
Password Hashing	argon2-cffi	latest	Secure password hashing
HTTP Client	httpx	latest	Async HTTP client

2.2 Frontend

Component	Technology	Version	Notes
Templates	Jinja2	(bundled)	Server-side rendering
Interactivity	HTMX	2.0+	CDN

Component	Technology	Version	Notes
Styling	Tailwind CSS	3.4+	CDN or CLI build
Icons	Heroicons	2.0	SVG icons
Multi-select	Tom Select	2.3+	CDN
Drag-and-drop	Sortable.js	1.15+	CDN
Date picker	Flatpickr	4.6+	CDN

2.3 Development Tools

Tool	Purpose
Ruff	Linting and formatting
mypy	Type checking
pytest	Testing
pytest-asyncio	Async test support

3. PROJECT STRUCTURE

```
playlist-service/
├── app/
│   ├── __init__.py
│   ├── main.py              # FastAPI app with lifespan
│   ├── config.py            # Pydantic settings
│   ├── dependencies.py          # FastAPI dependencies
│   ├── exceptions.py          # Custom exceptions
│   │
│   ├── models/              # SQLAlchemy ORM models
│   │   ├── __init__.py
│   │   ├── base.py           # Base, TimestampMixin
│   │   ├── admin.py
│   │   ├── group.py
│   │   ├── channel.py
│   │   ├── package.py
│   │   ├── tariff.py
│   │   └── user.py
│   │
```

```
│   ├── schemas/                # Pydantic schemas
│   │   ├── __init__.py
│   │   ├── common.py           # Response wrappers, pagination
│   │   ├── auth.py
│   │   ├── dashboard.py
│   │   ├── group.py
│   │   ├── channel.py
│   │   ├── package.py
│   │   ├── tariff.py
│   │   └── user.py
│   │
│   ├── services/               # Business logic
│   │   ├── __init__.py
│   │   ├── database.py         # Engine, session factory
│   │   ├── admin_auth.py       # Admin authentication
│   │   ├── group_service.py
│   │   ├── channel_service.py
│   │   ├── channel_sync.py     # Flussonic sync
│   │   ├── package_service.py
│   │   ├── tariff_service.py
│   │   ├── user_service.py     # Includes channel resolution
│   │   ├── playlist_generator.py
│   │   └── auth_sync.py        # Auth Service sync
│   │
│   ├── routes/                 # API endpoints
│   │   ├── __init__.py
│   │   ├── auth.py
│   │   ├── dashboard.py
│   │   ├── channels.py
│   │   ├── groups.py
│   │   ├── packages.py
│   │   ├── tariffs.py
│   │   ├── users.py
│   │   ├── lookup.py
│   │   └── pages.py            # HTML page routes
│   │
│   ├── clients/                # External API clients
│   │   ├── __init__.py
│   │   ├── flussonic.py
│   │   └── auth_service.py
│   │
│   ├── templates/              # Jinja2 templates
│   │   ├── base.html
│   │   ├── components/
│   │   │   ├── nav.html
│   │   │   ├── toast.html
│   │   │   ├── modal.html
```

```
│   │   │   ├── pagination.html
│   │   │   └── confirm_dialog.html
│   │   ├── auth/
│   │   │   └── login.html
│   │   ├── dashboard/
│   │   │   └── index.html
│   │   ├── channels/
│   │   │   ├── list.html
│   │   │   ├── _row.html
│   │   │   └── _edit_modal.html
│   │   ├── groups/
│   │   │   ├── list.html
│   │   │   ├── _row.html
│   │   │   └── _form.html
│   │   ├── packages/
│   │   │   ├── list.html
│   │   │   ├── _package_form.html
│   │   │   └── _tariff_form.html
│   │   └── users/
│   │       ├── list.html
│   │       ├── _row.html
│   │       ├── detail.html
│   │       └── _playlist_preview.html
│   │
│   ├── static/
│   │   ├── css/
│   │   │   └── app.css
│   │   └── js/
│   │       └── app.js
│   │
│   └── utils/
│       ├── __init__.py
│       ├── password.py          # hash_password, verify_password
│       ├── token.py             # generate_token
│       └── pagination.py        # Pagination helpers
│
├── alembic/
│   ├── versions/
│   ├── env.py
│   └── alembic.ini
│
├── scripts/
│   └── create_admin.py          # Create initial admin user
│
├── tests/
│   ├── __init__.py
│   ├── conftest.py
```

```
│   └── ...
│
├── .env.example
├── .gitignore
├── .dockerignore
├── pyproject.toml
├── Dockerfile
├── docker-compose.yml
└── README.md
```

---

4. CONFIGURATION

4.1 Environment Variables

Variable	Default	Description
`DATABASE_URL`	`postgresql+asyncpg://...`	Database connection string
`DB_POOL_SIZE`	`5`	Connection pool size
`DB_MAX_OVERFLOW`	`10`	Max overflow connections
`DB_POOL_TIMEOUT`	`30`	Pool timeout seconds
`DB_ECHO`	`false`	Log SQL queries
`SESSION_TIMEOUT`	`86400`	Admin session timeout (seconds)
`SECRET_KEY`	(required)	Secret key for sessions
`FLUSSONIC_URL`	(required)	Flussonic API base URL
`FLUSSONIC_API_KEY`	(required)	Flussonic API key
`AUTH_SERVICE_URL`	(required)	Auth Service base URL
`AUTH_SERVICE_API_KEY`	(required)	Auth Service API key
`API_HOST`	`0.0.0.0`	Server bind address
`API_PORT`	`8080`	Server port

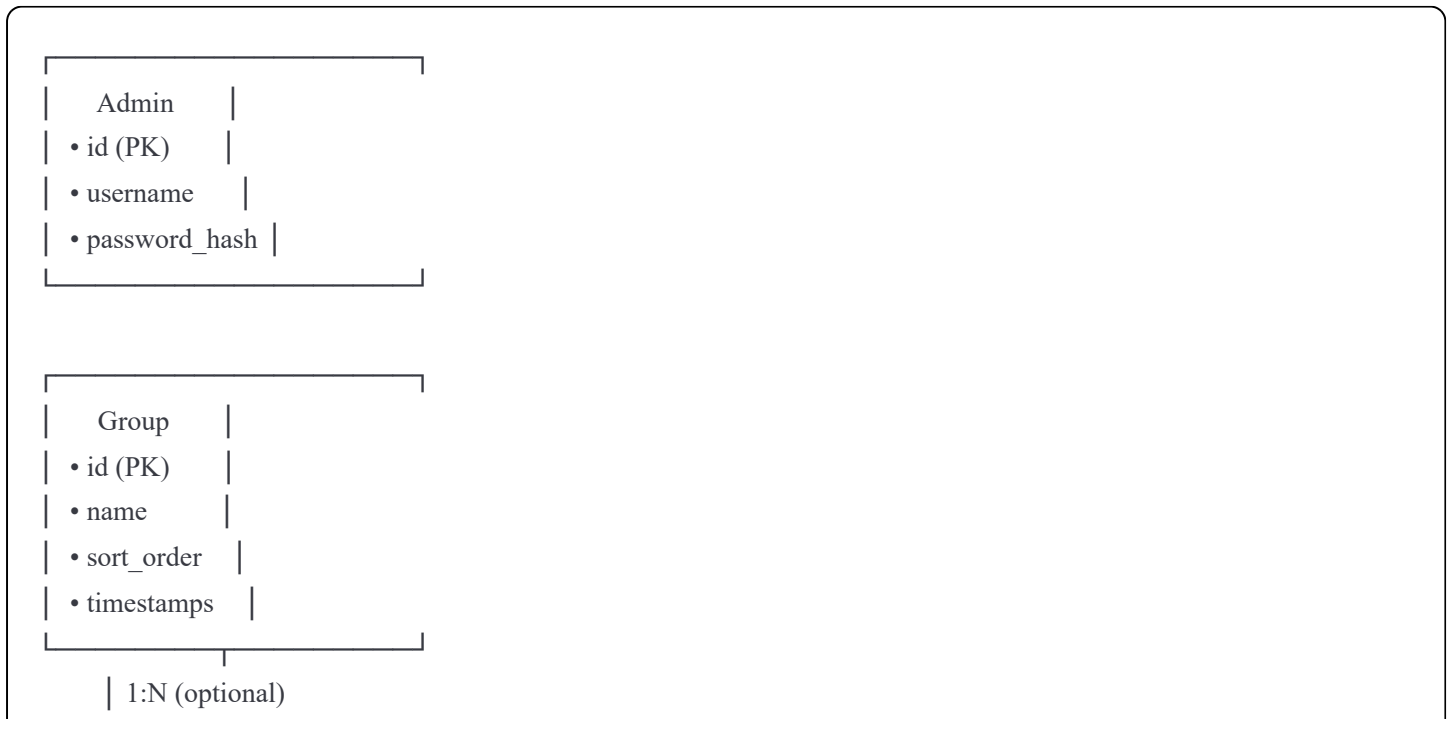
4.2 .env.example

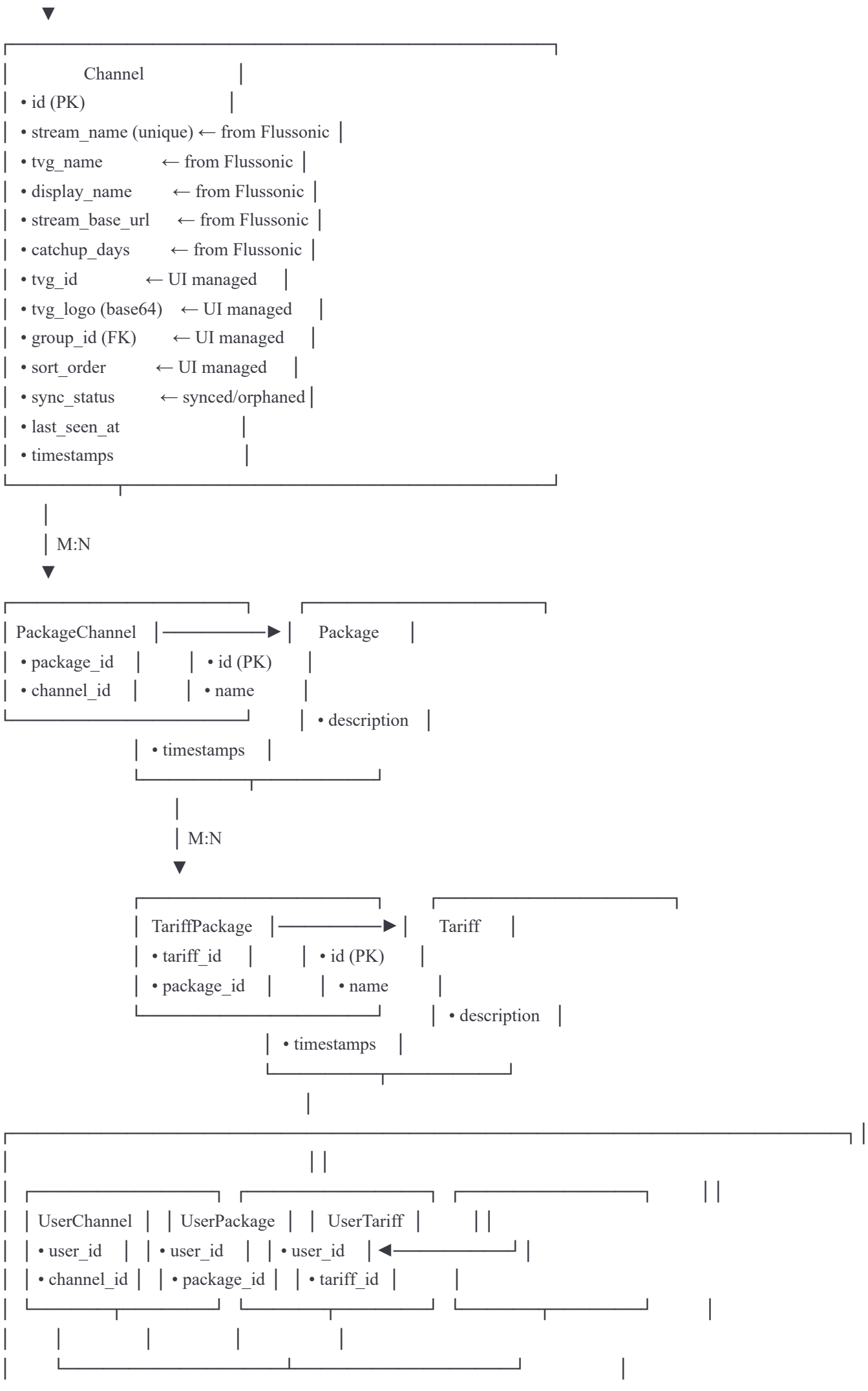
```
# Database
DATABASE_URL=postgresql+asyncpg://user:password@localhost:5432/playlist_service
DB_POOL_SIZE=5
DB_MAX_OVERFLOW=10
DB_ECHO=false

# Security
SECRET_KEY=your-secret-key-here
SESSION_TIMEOUT=86400

# Flussonic
FLUSSONIC_URL=http://flussonic-server:8080
FLUSSONIC_API_KEY=your-flussonic-api-key

# Auth Service
AUTH_SERVICE_URL=http://auth-service:8090
AUTH_SERVICE_API_KEY=your-auth-service-api-key

# Server
API_HOST=0.0.0.0
API_PORT=8080
```

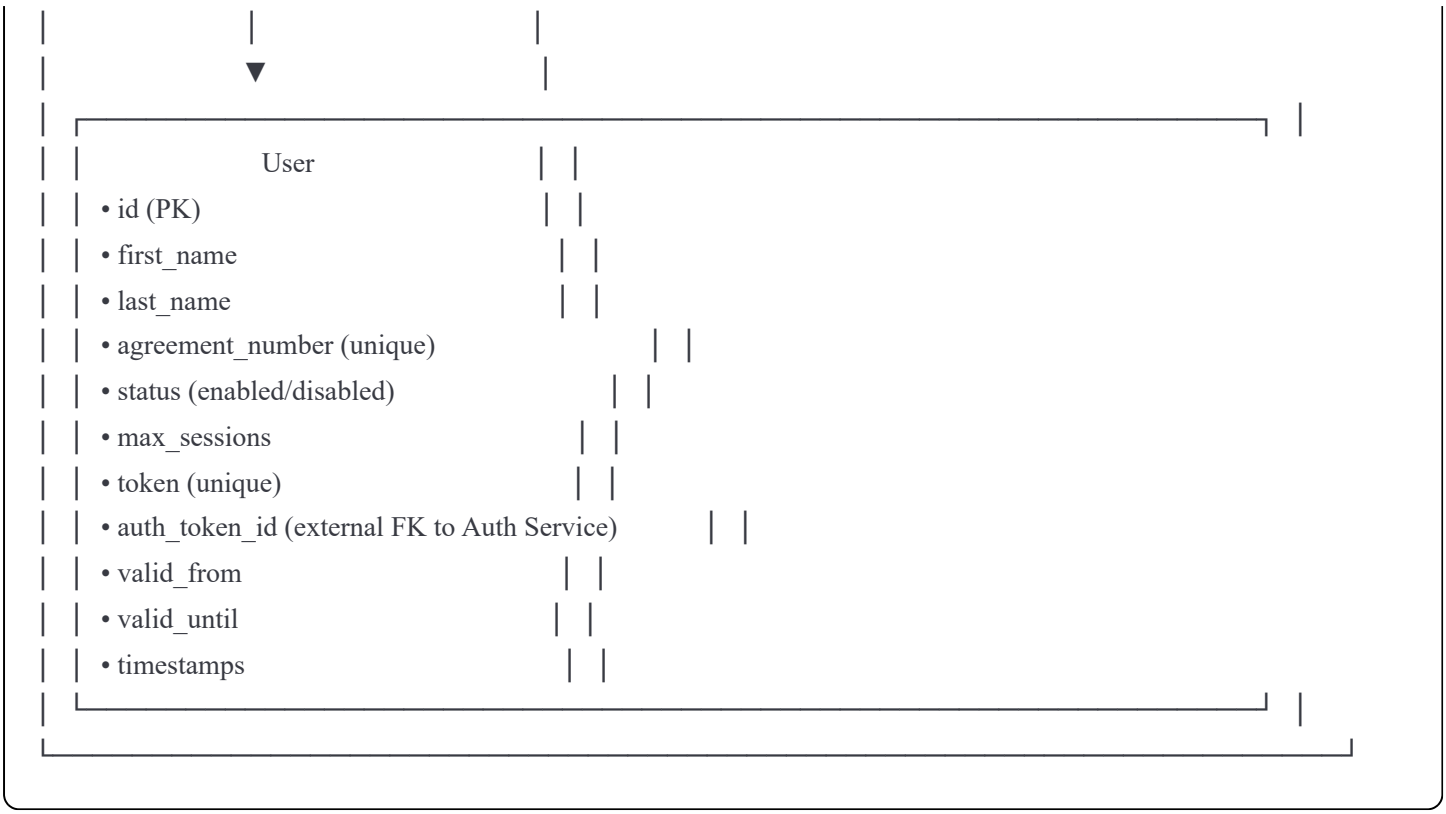
---

5. DATABASE SCHEMA

5.1 Entity Relationship Diagram




Channel
• id (PK)
• stream_name (unique) ← from Flussonic
• tvg_name ← from Flussonic
• display_name ← from Flussonic
• stream_base_url ← from Flussonic
• catchup_days ← from Flussonic
• tvg_id ← UI managed
• tvg_logo (base64) ← UI managed
• group_id (FK) ← UI managed
• sort_order ← UI managed
• sync_status ← synced/orphaned
• last_seen_at
• timestamps
M:N
PackageChannel
• package_id
• channel_id
Package
• id (PK)
• name
• description
• timestamps
M:N
TariffPackage
• tariff_id
• package_id
Tariff
• id (PK)
• name
• description
• timestamps
UserChannel
• user_id
• channel_id
UserPackage
• user_id
• package_id
UserTariff
• user_id
• tariff_id



5.2 Table Specifications

admins

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment
username	VARCHAR(100)	UNIQUE, NOT NULL, INDEX	Admin login
password_hash	VARCHAR(255)	NOT NULL	Argon2 hash

groups

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment
name	VARCHAR(100)	UNIQUE, NOT NULL	Group name
sort_order	INTEGER	DEFAULT 0, INDEX	Playlist ordering
created_at	TIMESTAMP	DEFAULT now()	Creation time
updated_at	TIMESTAMP	DEFAULT now()	Update time

channels

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment
stream_name	VARCHAR(255)	UNIQUE, NOT NULL, INDEX	Flussonic stream ID
tvg_name	VARCHAR(255)	NULLABLE	EPG name (Flussonic)
display_name	VARCHAR(255)	NULLABLE	Display name (Flussonic)
stream_base_url	VARCHAR(500)	NOT NULL	Stream URL (Flussonic)
catchup_days	INTEGER	NULLABLE	DVR days (Flussonic)
tvg_id	VARCHAR(100)	NULLABLE, INDEX	EPG ID (UI)
tvg_logo	TEXT	NULLABLE	Base64 logo (UI)
group_id	INTEGER	FK→groups.id, SET NULL	Group (UI)
sort_order	INTEGER	DEFAULT 0, INDEX	Order (UI)
sync_status	ENUM	DEFAULT 'synced', INDEX	synced/orphaned
last_seen_at	TIMESTAMP	NULLABLE	Last sync time
created_at	TIMESTAMP	DEFAULT now()	Creation time
updated_at	TIMESTAMP	DEFAULT now()	Update time

packages

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment
name	VARCHAR(100)	UNIQUE, NOT NULL	Package name
description	TEXT	NULLABLE	Description
created_at	TIMESTAMP	DEFAULT now()	Creation time
updated_at	TIMESTAMP	DEFAULT now()	Update time

package_channels (M:N)

Column	Type	Constraints
package_id	INTEGER	PK, FK→packages.id, CASCADE
channel_id	INTEGER	PK, FK→channels.id, CASCADE

tariffs

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment
name	VARCHAR(100)	UNIQUE, NOT NULL	Tariff name
description	TEXT	NULLABLE	Description
created_at	TIMESTAMP	DEFAULT now()	Creation time
updated_at	TIMESTAMP	DEFAULT now()	Update time

tariff_packages (M:N)

Column	Type	Constraints
tariff_id	INTEGER	PK, FK→tariffs.id, CASCADE
package_id	INTEGER	PK, FK→packages.id, CASCADE

users

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment
first_name	VARCHAR(100)	NOT NULL	First name
last_name	VARCHAR(100)	NOT NULL	Last name
agreement_number	VARCHAR(100)	UNIQUE, NOT NULL, INDEX	Contract ID
status	ENUM	DEFAULT 'enabled', INDEX	enabled/disabled
max_sessions	INTEGER	DEFAULT 1, NOT NULL	Max streams

Column	Type	Constraints	Description
token	VARCHAR(255)	UNIQUE, NOT NULL, INDEX	Auth token
auth_token_id	INTEGER	NULLABLE	Auth Service token ID
valid_from	TIMESTAMP	NULLABLE	Subscription start
valid_until	TIMESTAMP	NULLABLE	Subscription end
created_at	TIMESTAMP	DEFAULT now()	Creation time
updated_at	TIMESTAMP	DEFAULT now()	Update time

user_tariffs (M:N)

Column	Type	Constraints
user_id	INTEGER	PK, FK→users.id, CASCADE
tariff_id	INTEGER	PK, FK→tariffs.id, CASCADE

user_packages (M:N)

Column	Type	Constraints
user_id	INTEGER	PK, FK→users.id, CASCADE
package_id	INTEGER	PK, FK→packages.id, CASCADE

user_channels (M:N)

Column	Type	Constraints
user_id	INTEGER	PK, FK→users.id, CASCADE
channel_id	INTEGER	PK, FK→channels.id, CASCADE

5.3 Enums

SyncStatus: synced, orphaned
UserStatus: enabled, disabled

5.4 Cascade Rules

Parent	Child	On Delete
groups	channels.group_id	SET NULL
channels	package_channels	CASCADE
channels	user_channels	CASCADE
packages	package_channels	CASCADE
packages	tariff_packages	CASCADE
packages	user_packages	CASCADE
tariffs	tariff_packages	CASCADE
tariffs	user_tariffs	CASCADE
users	user_tariffs	CASCADE
users	user_packages	CASCADE
users	user_channels	CASCADE

---

6. BUSINESS LOGIC

6.1 Channel Sync Logic

Trigger: Manual (admin clicks "Sync from Flussonic" button)

Process:

1. Fetch all streams from Flussonic API
2. For each Flussonic channel:
    - If stream_name exists in DB → Update Flussonic fields only (preserve UI fields)
    - If stream_name not in DB → Create new channel with default sort_order
3. For channels in DB but not in Flussonic → Set sync_status = 'orphaned'
4. Return stats: total, new, updated, orphaned

Fields updated from Flussonic: tvg_name, display_name, stream_base_url, catchup_days

Fields preserved (UI-managed): tvg_id, tvg_logo, group_id, sort_order

6.2 Channel Resolution (for User)

User channels are resolved from three sources (union, deduplicated):

1. **Direct channels**: user_channels table
2. **Package channels**: user_packages → package_channels
3. **Tariff channels**: user_tariffs → tariff_packages → package_channels

Ordering: group.sort_order (nulls last), then channel.sort_order

6.3 Playlist Generation

Format: M3U8

Structure:

```
#EXTM3U
#EXTINF:-1 tvg-name="NAME" tvg-id="ID" catchup-days="N" group-title="GROUP" tvg-logo="LOGO",DISPLAY_NAME
http://BASE_URL/STREAM_NAME/video.m3u8?token=USER_TOKEN
```

Filename: `{last_name}_{first_name}_{agreement_number}.m3u8`

6.4 Auth Service Sync

On User Create:

1. Generate token
2. Resolve channels
3. POST to Auth Service /api/tokens with:
   - token, user_id, status, max_sessions
   - valid_from, valid_until
   - allowed_streams (resolved channel stream_names)
4. Store returned auth_token_id

On User Update:

1. Resolve channels
2. PATCH to Auth Service /api/tokens/{auth_token_id}

On User Delete:

1. DELETE from Auth Service /api/tokens/{auth_token_id}

6.5 Deletion Cascade Behavior

Channel (orphaned only):

- Show confirmation with affected packages/users count
- Remove from all packages and user assignments
- Delete channel

Group:

- Channels get group_id = NULL
- Show count of affected channels

Package:

- Remove from tariffs and users
- Show affected counts

Tariff:

- Remove from users
- Show affected count

User:

- Delete from Auth Service
- Remove all assignments

---

7. API SPECIFICATION

7.1 General

- **Base URL**: `/api/v1`
- **Content-Type**: `application/json`
- **Authentication**: Session cookie (`session_id`)

7.2 Response Formats

Success:

```
json

{"success": true, "data": {...}}
```

Error:

```
json

{"success": false, "error": {"code": "ERROR_CODE", "message": "..."}}
```

Paginated:

```
json

{"success": true, "data": {"items": [...], "total": 100, "page": 1, "per_page": 20, "pages": 5}}
```

7.3 Endpoints

Authentication

Method	Endpoint	Description
POST	/api/v1/auth/login	Login (username, password) → sets session cookie
POST	/api/v1/auth/logout	Logout → clears session
GET	/api/v1/auth/me	Get current admin info

Dashboard

Method	Endpoint	Description
GET	/api/v1/dashboard/stats	Get counts (channels, groups, packages, tariffs, users, last_sync)

Channels

Method	Endpoint	Description
GET	/api/v1/channels	List (page, per_page, search, group_id, sync_status, sort_by, sort_dir)
GET	/api/v1/channels/{id}	Get single channel
PATCH	/api/v1/channels/{id}	Update (tvg_id, tvg_logo only)
DELETE	/api/v1/channels/{id}	Delete orphaned channel
PATCH	/api/v1/channels/{id}/group	Update group assignment (group_id)
PATCH	/api/v1/channels/{id}/packages	Update package assignments (package_ids[])
POST	/api/v1/channels/reorder	Reorder (order: [{id, sort_order}])
POST	/api/v1/channels/sync	Trigger Flussonic sync
GET	/api/v1/channels/{id}/cascade-info	Get cascade delete info

Groups

Method	Endpoint	Description
GET	/api/v1/groups	List all groups
POST	/api/v1/groups	Create (name)
PATCH	/api/v1/groups/{id}	Update (name)
DELETE	/api/v1/groups/{id}	Delete
POST	/api/v1/groups/reorder	Reorder

Packages

Method	Endpoint	Description
GET	/api/v1/packages	List all packages
POST	/api/v1/packages	Create (name, description)
PATCH	/api/v1/packages/{id}	Update
DELETE	/api/v1/packages/{id}	Delete

Tariffs

Method	Endpoint	Description
GET	/api/v1/tariffs	List all tariffs
POST	/api/v1/tariffs	Create (name, description, package_ids[])
PATCH	/api/v1/tariffs/{id}	Update (including package_ids[])
DELETE	/api/v1/tariffs/{id}	Delete

Users

Method	Endpoint	Description
GET	/api/v1/users	List (page, search, status, tariff_id)
GET	/api/v1/users/{id}	Get single user
POST	/api/v1/users	Create (syncs to Auth Service)
PATCH	/api/v1/users/{id}	Update (syncs to Auth Service)
DELETE	/api/v1/users/{id}	Delete (removes from Auth Service)
POST	/api/v1/users/{id}/regenerate-token	Regenerate token
GET	/api/v1/users/{id}/resolved-channels	Get resolved channel list
GET	/api/v1/users/{id}/playlist	Download M3U file
GET	/api/v1/users/{id}/playlist/preview	Preview playlist content

Lookups (for dropdowns)

Method	Endpoint	Description
GET	/api/v1/lookup/groups	All groups (id, name)
GET	/api/v1/lookup/packages	All packages (id, name)
GET	/api/v1/lookup/tariffs	All tariffs (id, name)
GET	/api/v1/lookup/channels	Channels (search, limit)

7.4 Error Codes

Code	HTTP	Description
INVALID_CREDENTIALS	401	Invalid login
UNAUTHORIZED	401	Not authenticated
NOT_FOUND	404	Resource not found
VALIDATION_ERROR	422	Validation failed
DUPLICATE_ENTRY	409	Unique constraint violation
CHANNEL_NOT_ORPHANED	400	Can only delete orphaned channels
FLUSSONIC_ERROR	502	Flussonic API error
AUTH_SERVICE_ERROR	502	Auth Service error

---

8. UI SPECIFICATION

8.1 Pages Overview

Page	Route	Description
Login	/login	Admin login form
Dashboard	/	Stats, quick links, sync button
Channels	/channels	Channel list with inline editing
Groups	/groups	Group CRUD
Packages & Tariffs	/packages	Split view for packages and tariffs
Users	/users	User list
User Detail	/users/{id}	User edit form, playlist download

8.2 Login Page

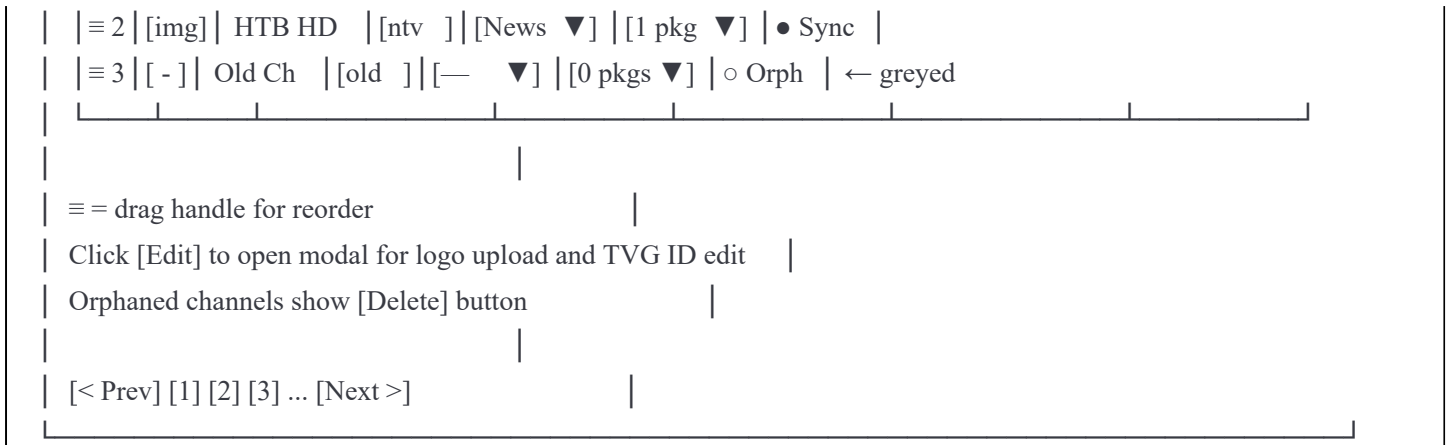
```
┌──────────────────────────────────────────────────────┐
│          Playlist Service                            │
│                                                      │
│  Username: [_____________________]                   │
│  Password: [_____________________]                   │
│                                                      │
│  [       Sign In       ]                             │
│                                                      │
│  (error message if login fails)                      │
└──────────────────────────────────────────────────────┘
```

8.3 Dashboard

```
┌──────────────────────────────────────────────────────────────────────────┐
│ Playlist Service                          [Admin ▼] Logout               │
├──────────────────────────────────────────────────────────────────────────┤
│ [Dashboard] [Channels] [Groups] [Packages & Tariffs] [Users]             │
├──────────────────────────────────────────────────────────────────────────┤
│                                                                          │
│  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ ┌──────────────┐     │
│  │  Channels    │ │   Groups     │ │  Packages    │ │   Users      │     │
│  │     124      │ │      12      │ │    8 / 3     │ │     256      │     │
│  │ [View All]   │ │ [View All]   │ │ [View All]   │ │ [View All]   │     │
│  └──────────────┘ └──────────────┘ └──────────────┘ └──────────────┘     │
│                                                                          │
│  Flussonic Sync                                                          │
│  Last sync: 2024-01-15 14:30          [Sync from Flussonic]              │
│  124 channels synced, 2 orphaned                                         │
│                                                                          │
└──────────────────────────────────────────────────────────────────────────┘
```

8.4 Channels Page

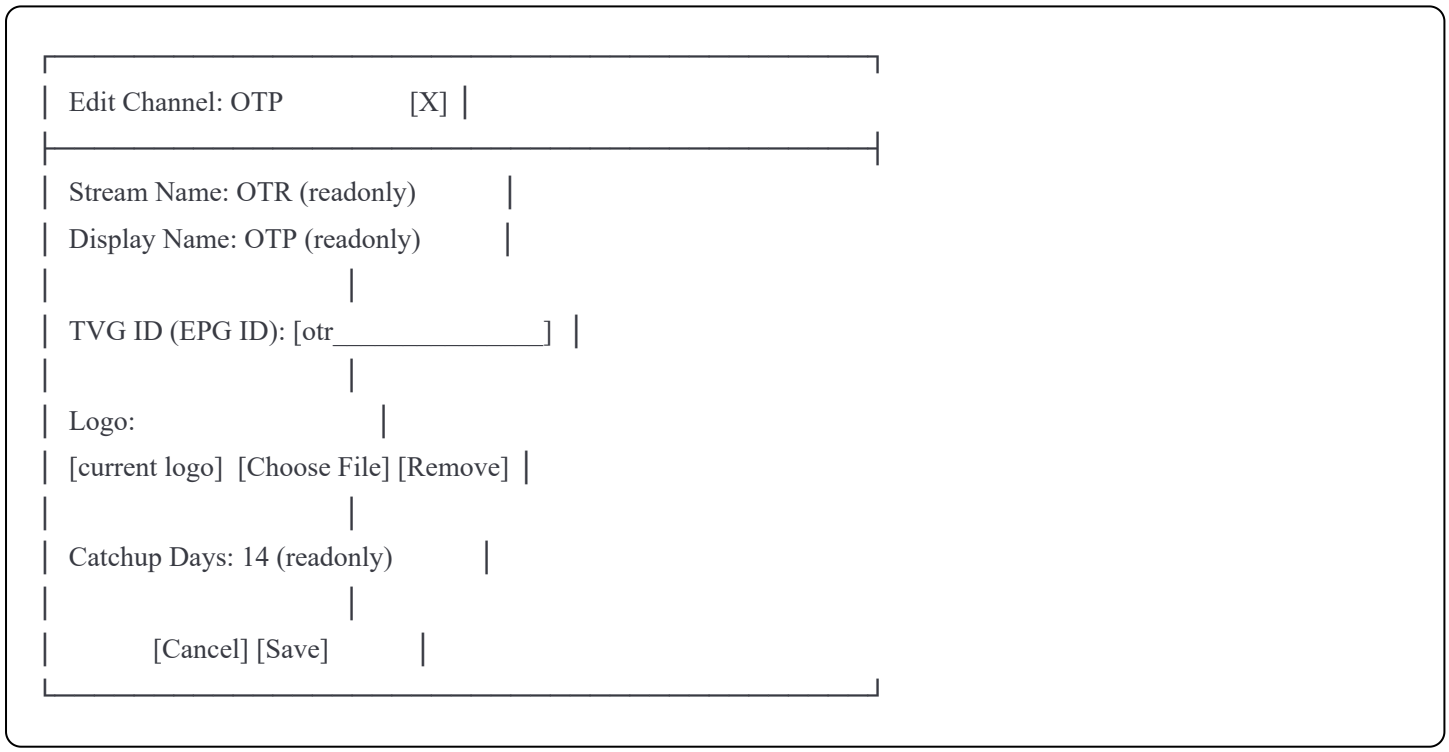
```
┌──────────────────────────────────────────────────────────────────────────┐
│ Channels                              [Sync from Flussonic]              │
├──────────────────────────────────────────────────────────────────────────┤
│ Filter: [All Groups ▼] [All Status ▼]  Search: [____________]            │
├──────────────────────────────────────────────────────────────────────────┤
│ │ # │Logo│ Name      │ TVG ID │ Group    │ Packages  │ Status │         │
│ ├───┼────┼───────────┼────────┼──────────┼───────────┼────────┤         │
│ │≡ 1│[img]│ OTP      │[otr   ]│[News  ▼] │[2 pkgs ▼] │● Sync  │         │
```



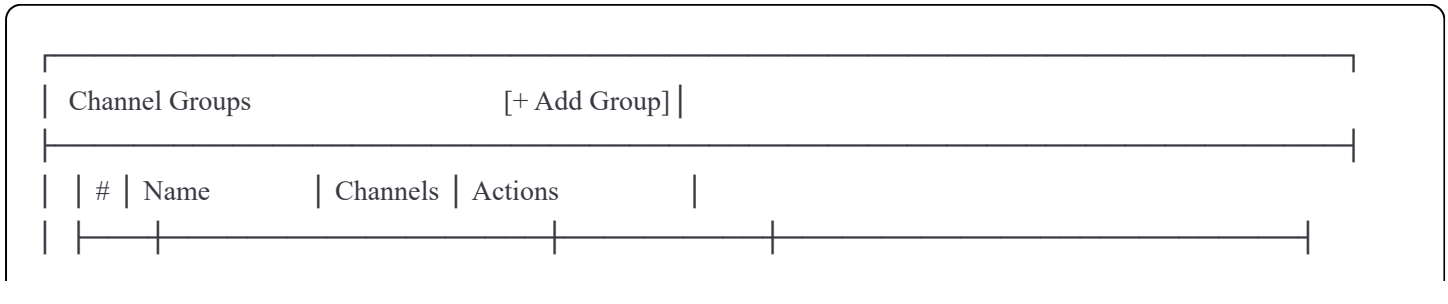
Inline editing:

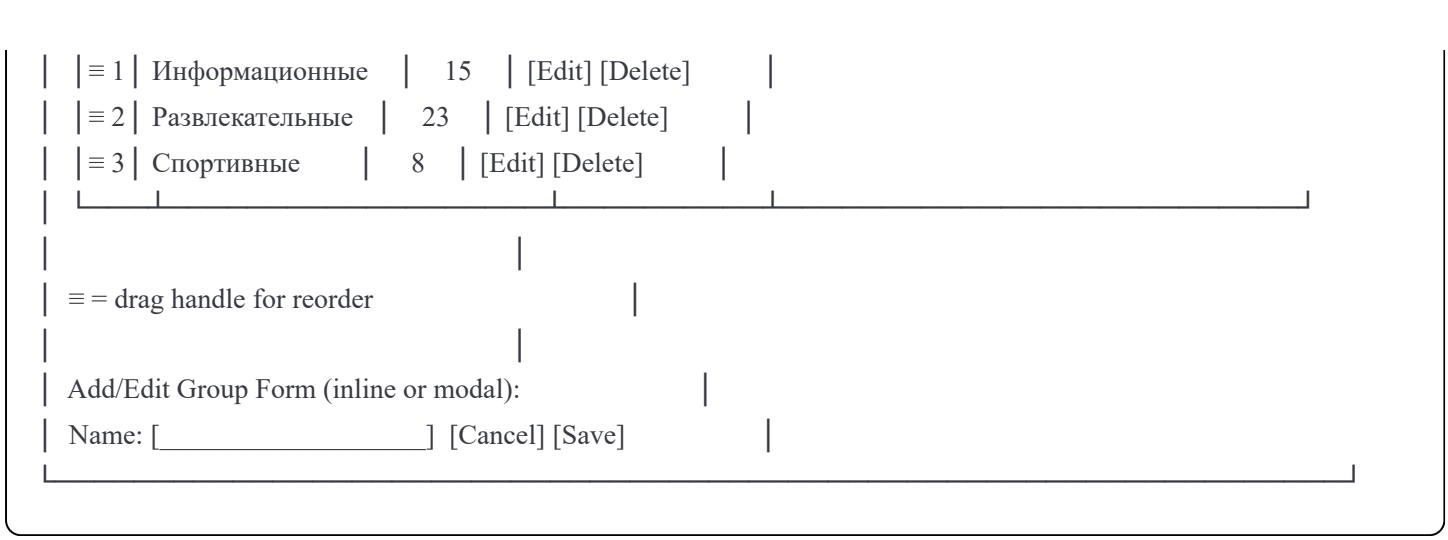
- Group dropdown: changes immediately via HTMX
- Packages multi-select: changes immediately via HTMX
- TVG ID: editable in modal
- Logo: uploadable in modal (stored as base64)

Channel Edit Modal:

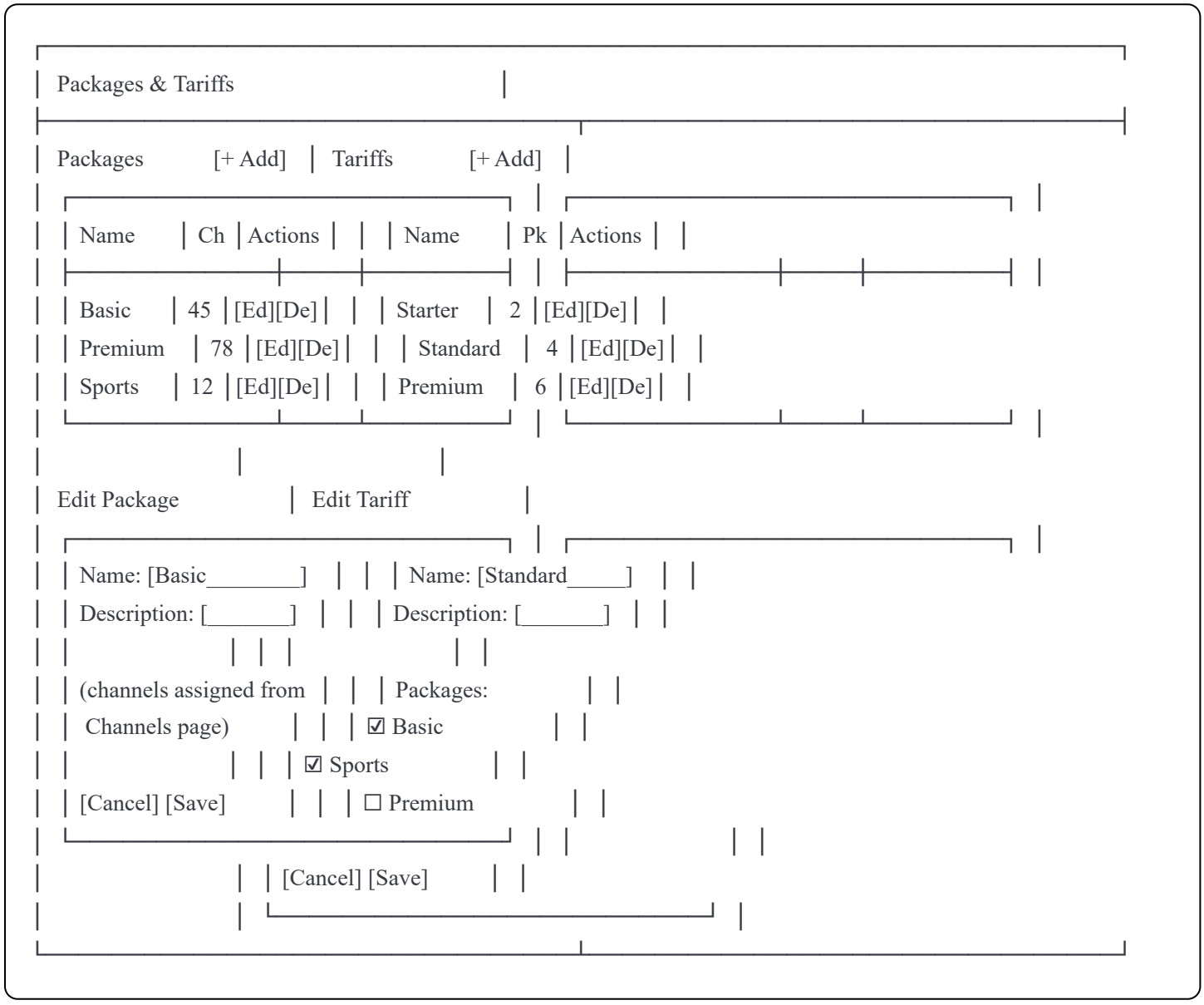


8.5 Groups Page



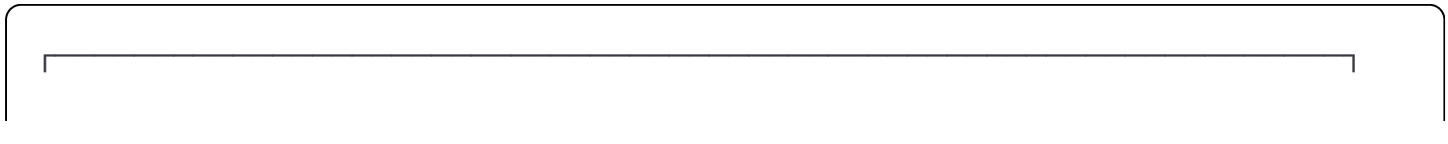


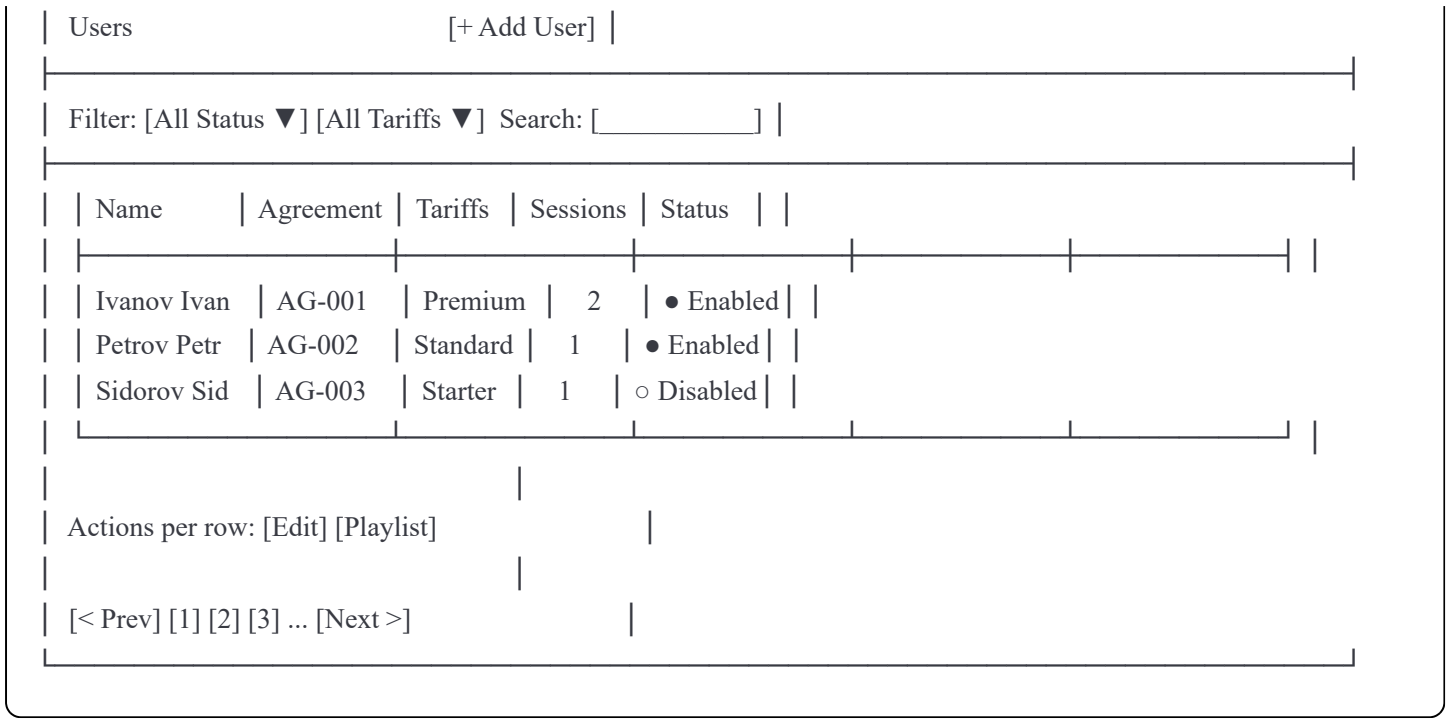
8.6 Packages & Tariffs Page (Split View)



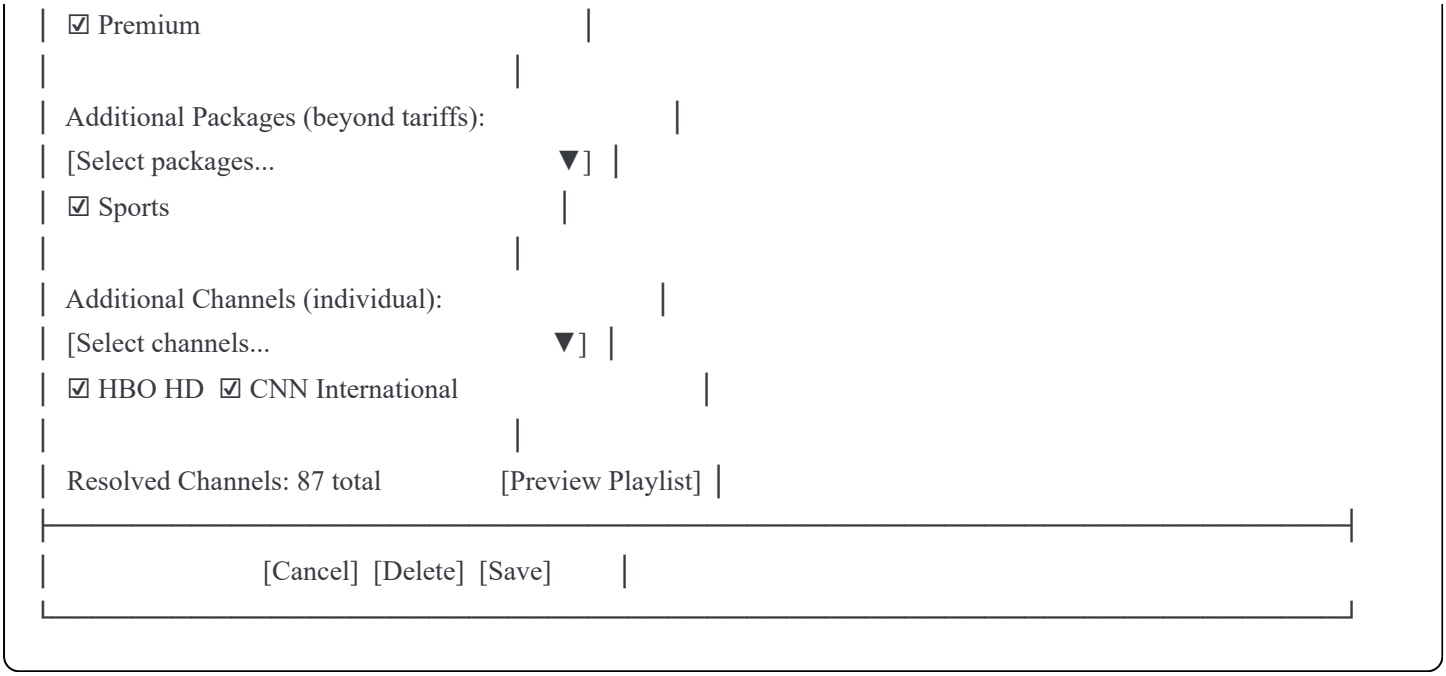
Note: Packages do NOT show channel list - channels are assigned FROM the Channels page.

8.7 Users List Page



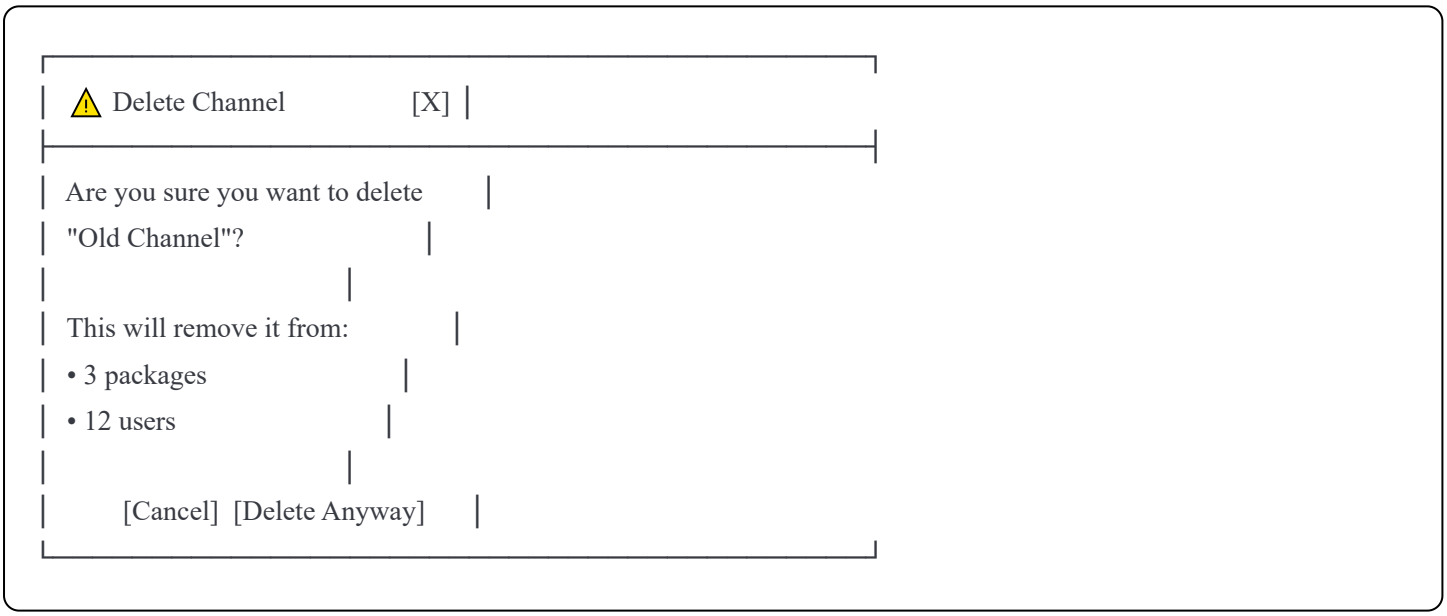


8.8 User Detail/Edit Page



8.9 Confirmation Dialogs

Delete confirmation with cascade warning:



8.10 Toast Notifications

Position: Top-right corner
Types: Success (green), Error (red), Warning (yellow)

---

9. EXTERNAL INTEGRATIONS

9.1 Flussonic API

Purpose: Fetch channel list (read-only)

Base URL: Configured via `FLUSSONIC_URL`

Authentication: API key in header or query param (depends on Flussonic version)

Required Endpoint:

- GET /streams or /api/v3/streams - List all streams

Expected Response Fields (per stream):

- name / stream_name - Unique stream identifier
- title / display_name - Human-readable name
- dvr / catchup_days - DVR days available
- Base URL for streaming

Note: Exact endpoint and response format depends on Flussonic version. Implementation should be adaptable.

9.2 Auth Service API

Purpose: Register and manage user tokens

Base URL: Configured via `AUTH_SERVICE_URL`

Authentication: `X-API-Key` header with `AUTH_SERVICE_API_KEY`

Endpoints Used

POST /api/tokens - Create token

```
json
```

Request:

```
{
  "token": "user-generated-token",
  "user_id": "user-agreement-number",
  "status": "active",
  "max_sessions": 2,
  "valid_from": "2024-01-01T00:00:00Z",
  "valid_until": "2024-12-31T23:59:59Z",
  "allowed_streams": ["stream1", "stream2", ...],
  "meta": {"first_name": "...", "last_name": "..."}
}
```

Response:

```
{
  "id": 123,  // ← Store this as auth_token_id
  "token": "...",
  ...
}
```

PATCH /api/tokens/{id} - Update token

json

Request (partial update):

```
{
  "status": "suspended",  // or "active"
  "max_sessions": 1,
  "allowed_streams": ["stream1", "stream2", ...]
}
```

DELETE /api/tokens/{id} - Delete token

Status Mapping

Playlist Service User Status	Auth Service Token Status
enabled	active
disabled	suspended

10. DEPLOYMENT

10.1 Docker

Dockerfile requirements:

- Python 3.12+ base image
- Install uv package manager
- Copy application code
- Install dependencies via uv
- Run with uvicorn

docker-compose.yml:

- playlist-service container
- PostgreSQL container (or external)
- Network shared with auth-service
- Volume for persistent data
- Health checks

10.2 Initial Setup

1. Create database
2. Run Alembic migrations
3. Create admin user via script
4. Configure environment variables
5. Start service

10.3 Admin Creation Script

Script at `scripts/create_admin.py`:

- Accepts username as argument or prompt
- Prompts for password (with confirmation)
- Creates admin record with hashed password

APPENDIX A: Sample Playlist Output

```
#EXTM3U
#EXTINF:-1 tvg-name="OTP" tvg-id="otr" catchup-days="14" group-title="Информационные" tvg-logo="...",OTP
http://stream.rutvportal.com:8085/OTR/video.m3u8?token=QeHUZDtsuc56bX69kLWk9Q
#EXTINF:-1 tvg-name="9 канал HD" tvg-id="2672" catchup-days="14" group-title="Израильские" tvg-logo="...",9 канал HD
http://stream.rutvportal.com:8085/9kanalHD/video.m3u8?token=QeHUZDtsuc56bX69kLWk9Q
```

APPENDIX B: Implementation Checklist

Phase 1: Foundation

- [ ] Project setup (pyproject.toml, dependencies)
- [ ] Configuration (config.py, .env)
- [ ] Database setup (engine, session factory)
- [ ] All SQLAlchemy models
- [ ] Alembic initial migration

Phase 2: Core Services

- [ ] Password utilities
- [ ] Token generation utilities
- [ ] Admin authentication service
- [ ] Group service
- [ ] Channel service
- [ ] Package service
- [ ] Tariff service
- [ ] User service (with channel resolution)
- [ ] Playlist generator service

Phase 3: External Integrations

- [ ] Flussonic API client
- [ ] Channel sync service
- [ ] Auth Service API client
- [ ] Auth sync service

Phase 4: API Layer

- [ ] Common schemas
- [ ] All domain schemas
- [ ] FastAPI dependencies
- [ ] All route handlers

Phase 5: UI Layer

- [ ] Base template with navigation
- [ ] Reusable components
- [ ] Login page
- [ ] Dashboard page
- [ ] Channels page with partials
- [ ] Groups page with partials
- [ ] Packages & Tariffs page with partials
- [ ] Users pages with partials

Phase 6: Finalization

- [ ] Page routes (HTML serving)
- [ ] Main application assembly
- [ ] Docker configuration
- [ ] README documentation
- [ ] Admin creation script