



REST Code Patterns

How to write less (boilerplate) code



PAOLO ROSSI
WINTech ITALIA **CTO**

SENCHA & EMB



MVP





GITHUB PROJECTS



github.com/paolo-rossi



Delphi JWT

JSON Web Token Library



WiRL

REST Library for Delphi



Linux Daemon

Real Linux daemons



Delphi Neon

JSON Serialization Library



OpenAPI-Delphi

OpenAPI 3.0 Library



NATS Delphi

NATS Client Library for Delphi



AGENDA

- How to free stuff
- Auth Server
- Async operations (Task Server)
- Automate CRUD operations
 - ◆ Entity to SQL
 - ◆ DataSet to Entity
 - ◆ Base class for CRUD resource
- Horizontal communication (NATS)



How to free stuff

How to free other objects from a resource method

1



FREEING STUFF

- Result is automatically freed in (almost) every framework
- WiRL also frees method parameters
- Remember [Singleton] to not free the result





USING THE OWNER

- When dealing with components
 - ◆ Useful with DataModules, Queries, Connections
- You can free anything: beware of the order!
- It's an effective but not a clean solution (IMHO)





CONTEXT INJECTION

- Used in WiRL from the start
- Used for “WiRL” objects
- Now it’s possible to register your own Context Factory
- Use it like other contexts -> [Context] Object: TMyClass





WIRL GARBAGE COLLECTOR

- Used to free objects
- Injectable with [Context]
- Assign to GC object or even records
- Used not only to free objects
 - ◆ Finalize objects, records, arrays, etc...





Auth Server

How to build a single auth server to manage... auth

2



WHY SINGLE AUTH?

- Not to duplicate auth-related code
 - ◆ It's difficult to update later
- To be sure to have always the latest version of auth
- Easier to change policies, database schema, encryption, hashing, etc...
- If you have a vulnerability, it spreads across your entire API ecosystem



HOW TO BUILD

- Use can use a different REST library
- You can use a different language! (e.g. golang)
- Identify the security requirements
- Identify the technology to use
 - ◆ JWT
 - ◆ SAML
 - ◆ OAuth2



AUTH SERVER

- The Auth server must authenticate the user and respond with a token (e.g. JWT)
 - ◆ That's it
- The token must be used to request from other services
 - ◆ The token can be issued for a peculiar service (or services)
 - ◆ Use the aud (audience) claim
 - ◆ The service must validate against the aud claim



AUTH SERVER

- The Auth server must authenticate the user and respond with a token (e.g. JWT). That's it
- The token must be used to request from other services
 - ◆ The token can be issued for a peculiar service (or services)
 - ◆ Use the aud (audience) claim
 - ◆ The (other) service must validate against the aud claim





Task Server

How to approach async operations

3



ASYNC OPERATIONS

- Long operations that don't return before request timeout
 - ◆ Very long queries
 - ◆ PDF or document generation
 - ◆ Batch operations
- It's not a good idea to have long requests
 - ◆ The client can decide to act: re-launches the request, closes the client...
- The server must respond with a 201 or 202 code
 - ◆ Created or Accepted
- And then... ?



ASYNC OPERATIONS

- The server responds with code and a response that contains the task identifier so the client can ask about it later
 - ◆ taskid: 125, status: accepted, estimated-time: 10min
- When the task is completed the server respond with something like:
 - ◆ taskid: 125, status: completed, link: /rest/taskserver/task/125
- Remember to set the Accept field accordingly



TASK SERVER

→ Several options

- ◆ No direct communication, only through database (TASK TABLE)
- ◆ The Original Server (REST) contacts the Task Server (REST)
- ◆ The client after the first (original) request get redirected to the Task Server (from the 201 response data)





CRUD

Automate CRUD operations

4



CRUD OPERATIONS



Create



Read



Update



Delete

C

R

U

D



CRUD OPERATIONS

GetCollection	SELECT * FROM TABLE
GetItem	SELECT * FROM TABLE WHERE ID=:ID
InsertItem	INSERT INTO TABLE
UpdateItem	UPDATE TABLE
DeleteItem	DELETE FROM TABLE



ENTITIES & DB

- We don't need an ORM in REST
 - ◆ At least not a full featured ORM
 - ◆ We need the automatic loading from a class definition
 - ◆ We don't need the caching and automatic update mechanisms
- Example: load from database a TPerson class
 - ◆ Generate the SQL from TPerson (properties)
 - ◆ Execute the SQL
 - ◆ Load data from TDataSet to a TPerson object



ENTITIES & DB

- For the insert & update we need
 - ◆ Read the JSON data considering the TPerson class
 - ◆ Generate the SQL from TPerson (properties)
 - ◆ Fill the SQL params with data from JSON
 - ◆ Execute the SQL
- You can go from there...



EDM ENGINE

- Introducing the EDM engine
- It's not an ORM
- It's not a library (probably will be a Neon extension)
- It's a sample code, you have to customize it





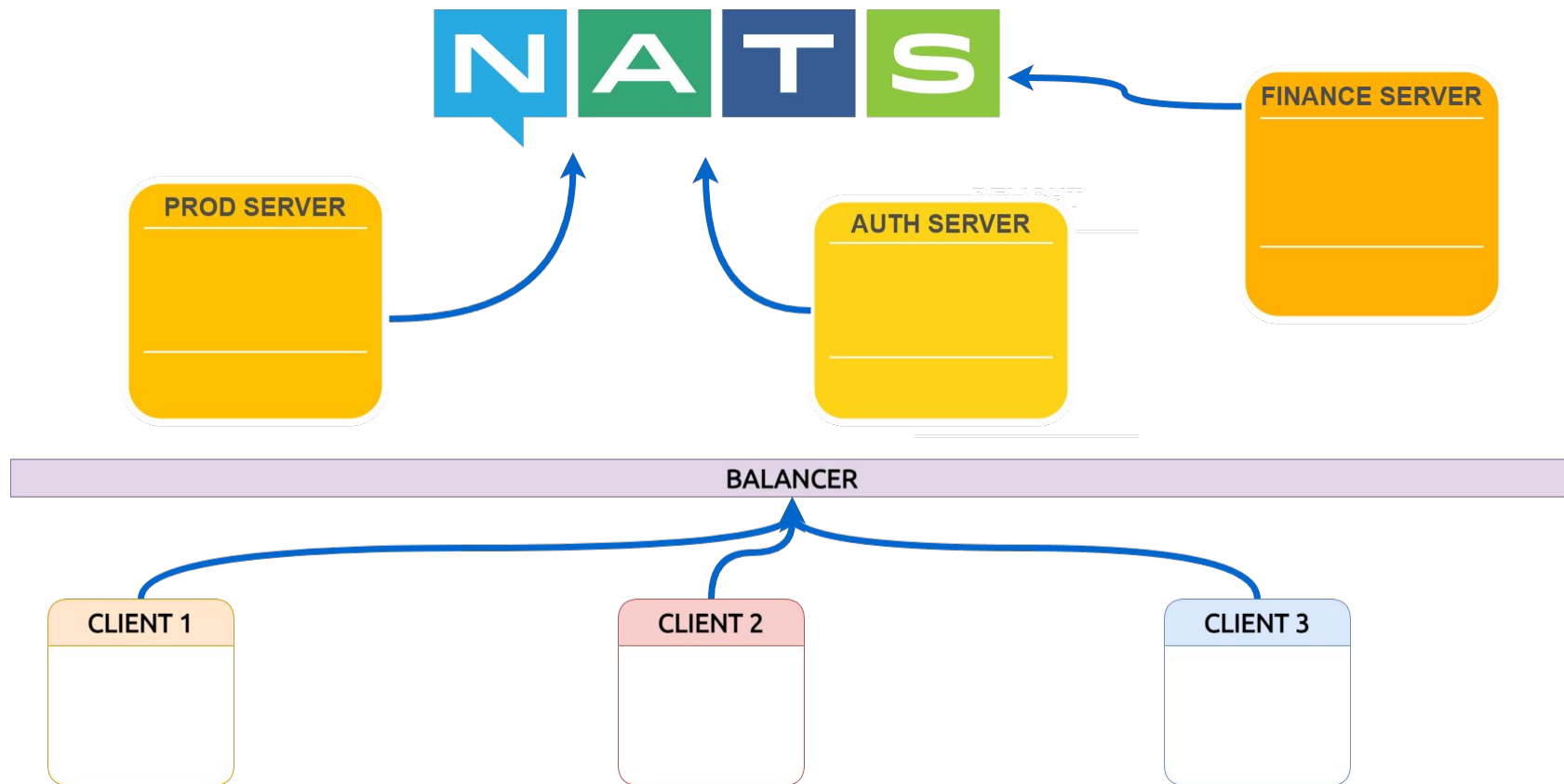
Messaging

Implement horizontal communication

5



HORIZONTAL MESSAGING





THANK YOU