



Managing your data with

# FastAPI & Piccolo Admin

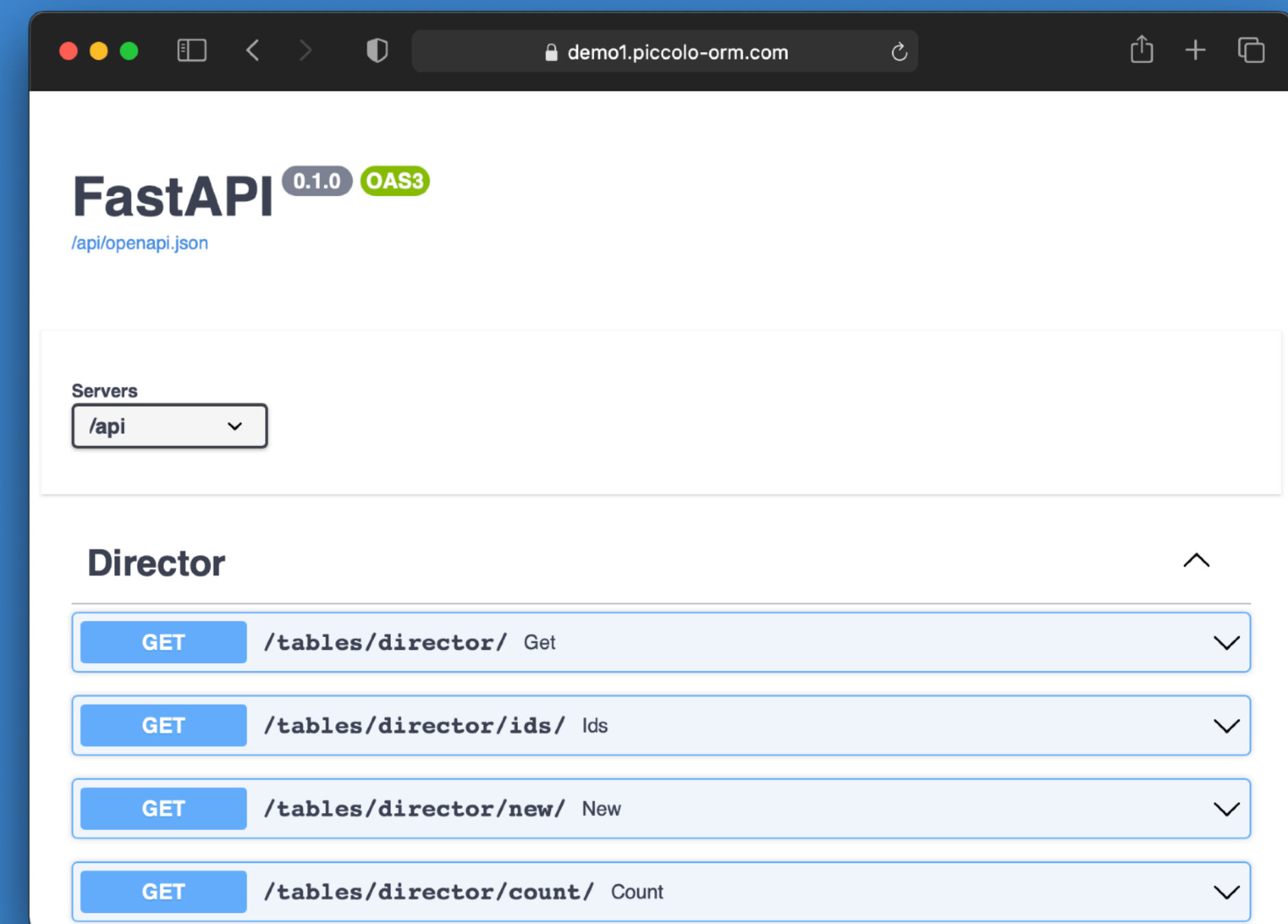




# What is FastAPI?

- A **modern web framework** which makes building **Swagger docs** for your **REST API** very simple.
- One of a new breed of **async** web frameworks based around the **ASGI** standard.
- The ‘**new hotness**’.

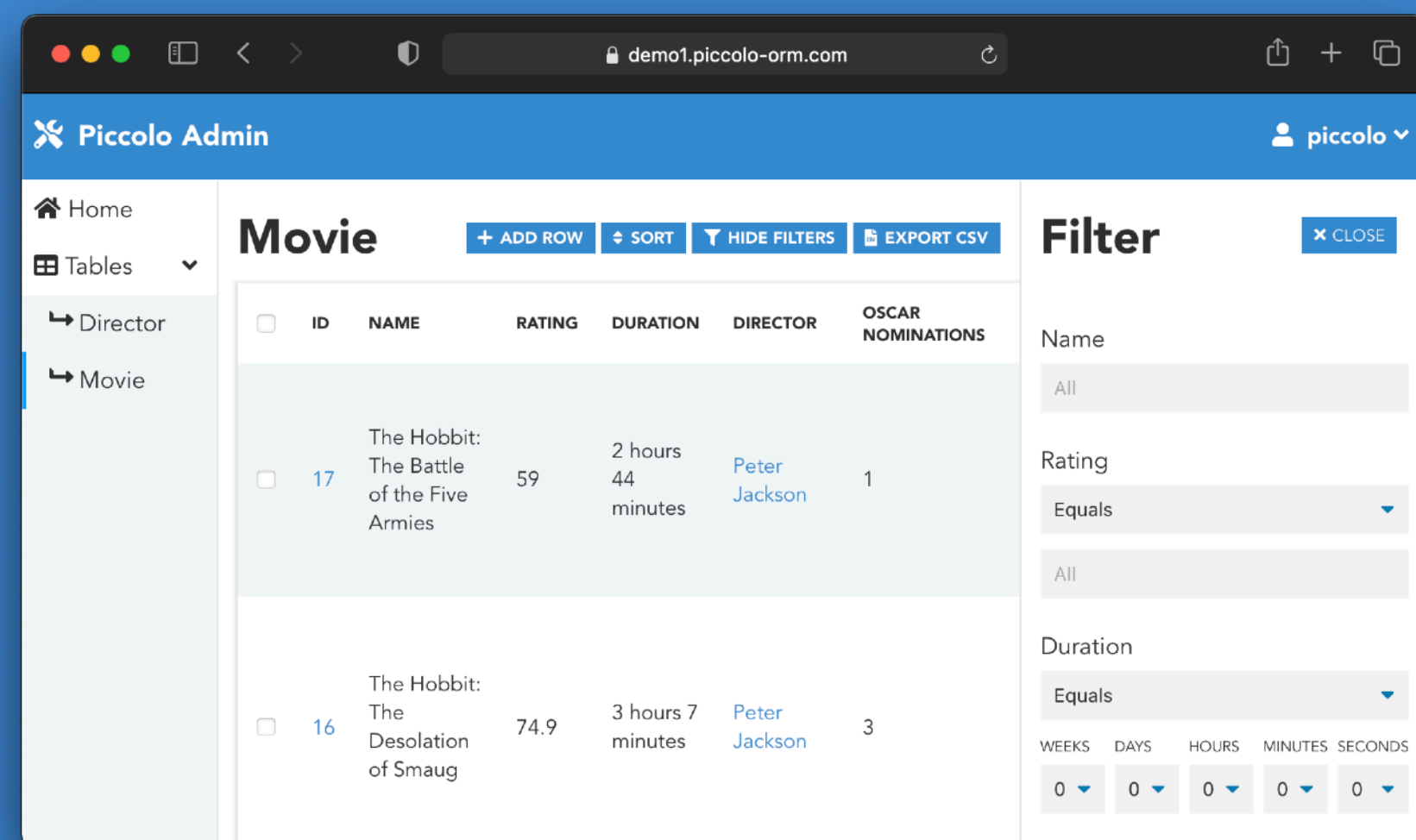
Swagger docs  
(also known as OpenAPI)





# What is **Piccolo**?

- An **async query builder** and **ORM**, with lots of **batteries included**.
- It works primarily with **PostgreSQL**.
- Integrates nicely with **FastAPI** and other ASGI frameworks.
- Built as a response to the lack of **async** ORMs / query builders at the time, and the desire for an **A+++ admin interface**.



← Piccolo Admin

Our good friend, PostgreSQL →





# Does **async** matter?

- The **async** / **await** keywords were added in **Python 3.6**, to make it easier to work with **asyncio**.
- **Asyncio** is an approach to **concurrency** based on an **event loop** and **coroutines**, and can be **more performant** than alternatives such as threads.
- Async is useful when your application is **I/O bound** - for example when **waiting for a database or API call**.
- It usually takes a few milliseconds to get a response from a database - even in Python land we can do useful work during that time.
- **Async isn't essential for all applications** - but it's a **great tool** to have in your tool belt.





# Why PostgreSQL?

- Piccolo mostly supports **PostgreSQL** (with some SQLite support).
- It was decided to provide **excellent support for one main database**, and to consider adding others in the future.
- PostgreSQL **continues to grow\***, and can support **diverse workloads** with it's **JSON**, **GIS** (via PostGIS), and **Time Series** (via TimescaleDB) support.



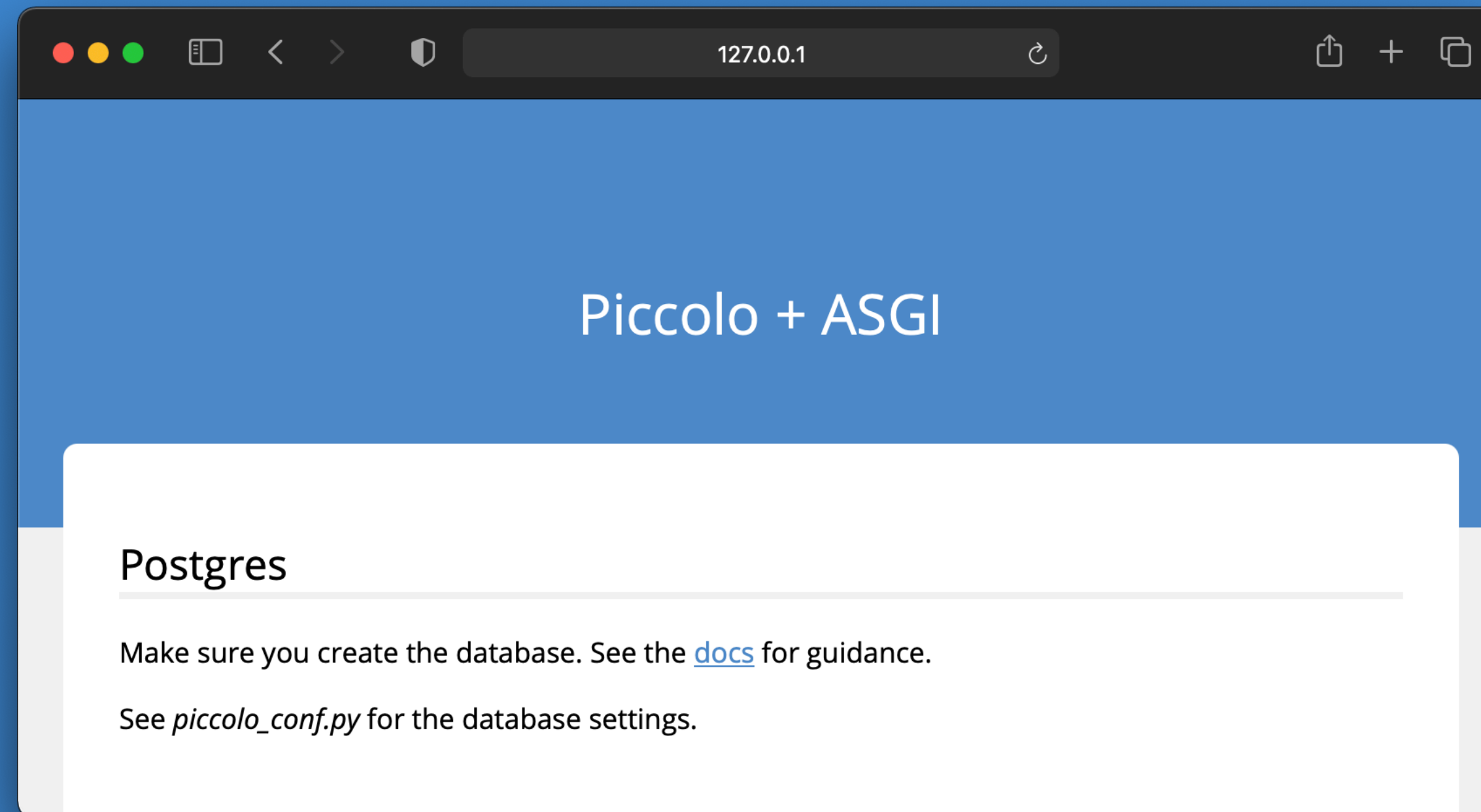
\* [https://db-engines.com/en/blog\\_post/85](https://db-engines.com/en/blog_post/85)



# Let's code an ASGI app

- A **FastAPI** and **Piccolo** app was scaffolded using:

```
>>> piccolo asgi new
```





# IMDb - meet PyMDb

- For this demo, we're going to look at a movie database we created, PyMDb!







# Schema

BaseUser	
id	Serial
username	Varchar
password	Secret
first_name	Varchar
last_name	Varchar
email	Varchar
active	Boolean
admin	Boolean
superuser	Boolean
last_login	Timestamp

SessionsBase	
id	Serial
token	Varchar
user_id	Integer
expiry_date	Timestamp
max_expiry_date	Timestamp

Studio	
id	Serial
name	Varchar
facilities	JSON

Review	
id	Serial
name	Varchar
review	Text
score	Integer
movie	ForeignKey



movie

Movie	
id	Serial
name	Varchar
rating	Real
duration	Interval
director	ForeignKey
oscar_nominations	Integer
won_oscar	Boolean
description	Text
release_date	Timestamp
box_office	Numeric
tags	Array
barcode	BigInt
genre	SmallInt



director

Director	
id	Serial
name	Varchar
years_nominated	Array
gender	Varchar



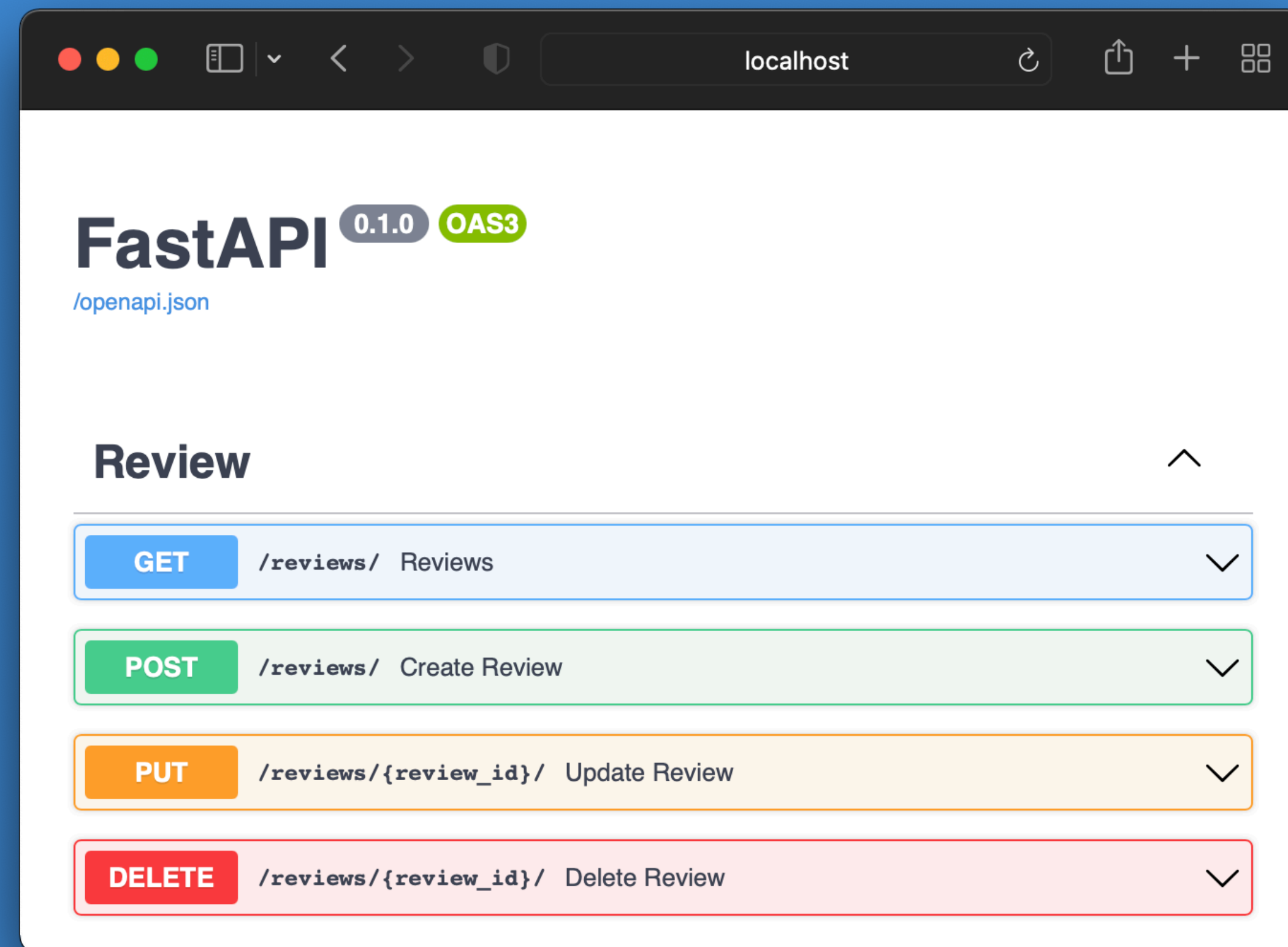
# Code tour

- Let's take a look at the code behind our database and app.
- We'll see:
  - What **Pydantic** is
  - How to create a **FastAPI** app and **endpoints**
  - How **FastAPIWrapper** can save us a lot of time
- You can see the full codebase here <https://github.com/piccolo-orm/pymdb>



# UI tour - Swagger docs

- Let's take a tour of the Swagger docs which FastAPI automatically creates.





# UI tour - Piccolo Admin

- Let's look at the powerful filtering capabilities of Piccolo admin.

The screenshot displays the Piccolo Admin web application. The interface includes a sidebar with navigation links for Home, Tables, Director, and Movie. The main content area shows a table of movies with columns for ID, NAME, RATING, DURATION, DIRECTOR, and OSCAR NOMINATION. A filter panel is open on the right, allowing users to filter the data by Name, Rating, and Duration. The filter panel also includes a 'CLOSE' button.

**Movie Table Data:**

ID	NAME	RATING	DURATION	DIRECTOR	OSCAR NOMINATION
17	The Hobbit: The Battle of the Five Armies	59	2 hours 44 minutes	Peter Jackson	1
16	The Hobbit: The Desolation of Smaug	74.9	3 hours 7 minutes	Peter Jackson	3

**Filter Panel:**

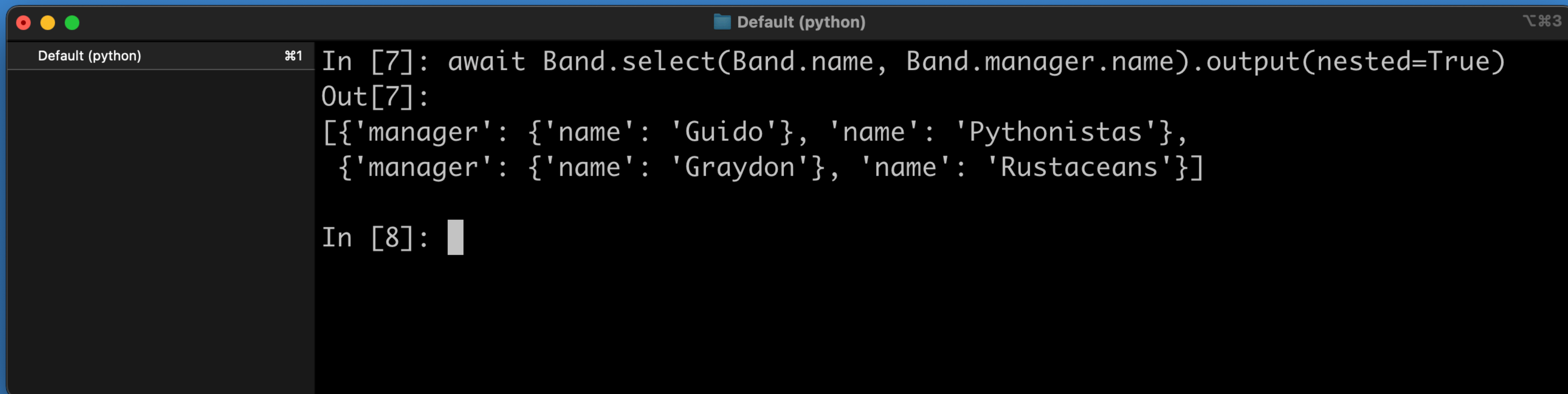
- Name: All
- Rating: Equals
- Duration: Equals
- WEEKS: 0
- DAYS: 0
- HOURS: 0
- MINUTES: 0
- SECONDS: 0



# Piccolo playground

- How about more complex queries?
- One of Piccolo's nicest features is the **playground** - which is a sandbox for **learning more about database querying**. Let's give it a go.

```
>>> piccolo playground run
```



```
Default (python)  %1 In [7]: await Band.select(Band.name, Band.manager.name).output(nested=True)
Out[7]:
[{'manager': {'name': 'Guido'}, 'name': 'Pythonistas'},
 {'manager': {'name': 'Graydon'}, 'name': 'Rustaceans'}]

In [8]:
```



# Other batteries

Piccolo has lots of other batteries included, and supports the following out of the box:

- Automatic database migrations
- Authentication (session auth and token auth)
- Custom CLI commands
- Automatic schema generation (piccolo schema generate)
- Schema visualisation (piccolo schema graph)
- And more!



# Get involved!

- All feedback is welcome
- Open to new contributors
- We're a friendly bunch on GitHub - [github.com/piccolo-orm](https://github.com/piccolo-orm)



# Thank you!

Any Questions?

GitHub: @dantownsend

Twitter: @danieltownsend

Email: [dan@dantownsend.co.uk](mailto:dan@dantownsend.co.uk)