Automating API development with JIT APIs

Dominic Hall- Supervised by: Konstantinos Markantonakis
Information Security Group, Smart Card and IoT Security Centre

Objectives

-Reduce development time by automating common development processes.
-Improve security and consistency of application code by generating clean and safe code.
-Encourage frontend driven development for better UX.
-Avoid increasing limitations and development time for non-automated functionality.

Introduction

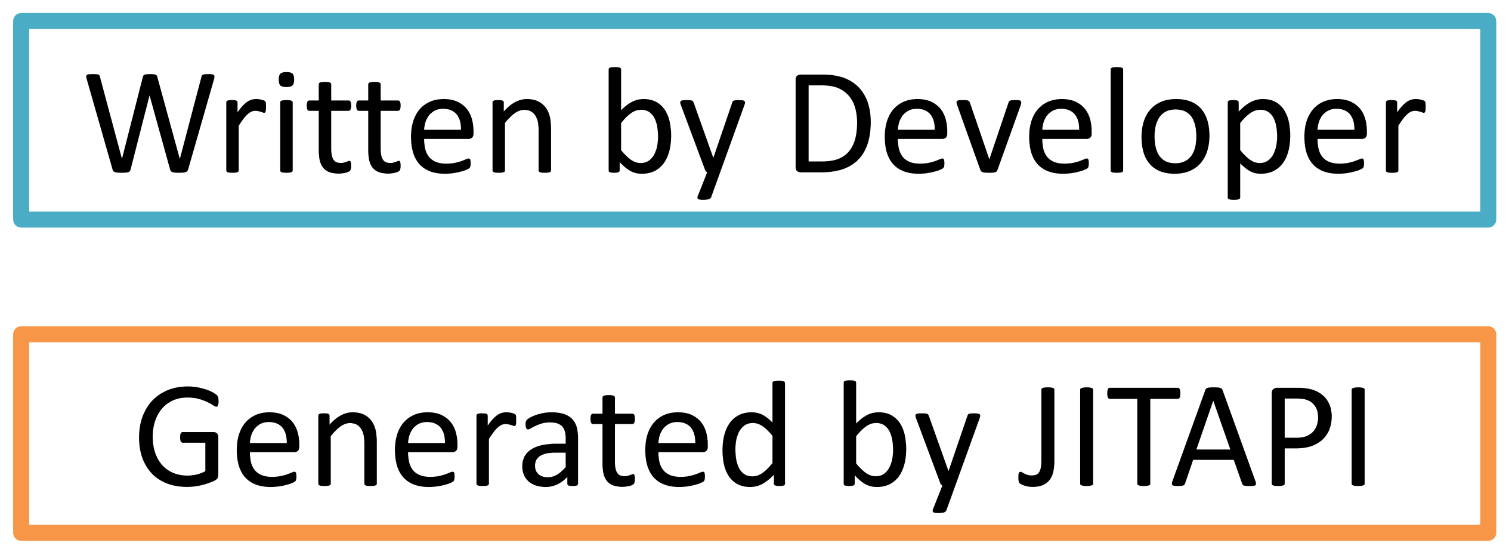
With the widespread use of REST APIs and other web API standards repeated writing of similar code has become commonplace. There are several advantages of automating this process.
-Reducing developer time
-Remove potential for convention and security slips (In generated code)
-Faster project setup time.
-Easier changes to backend code leads to less constrained frontend.

JIT Research Problems

As with any concept that looks this useful there are plenty of reasons why this has never beend one before. The current primary issues are as follows:

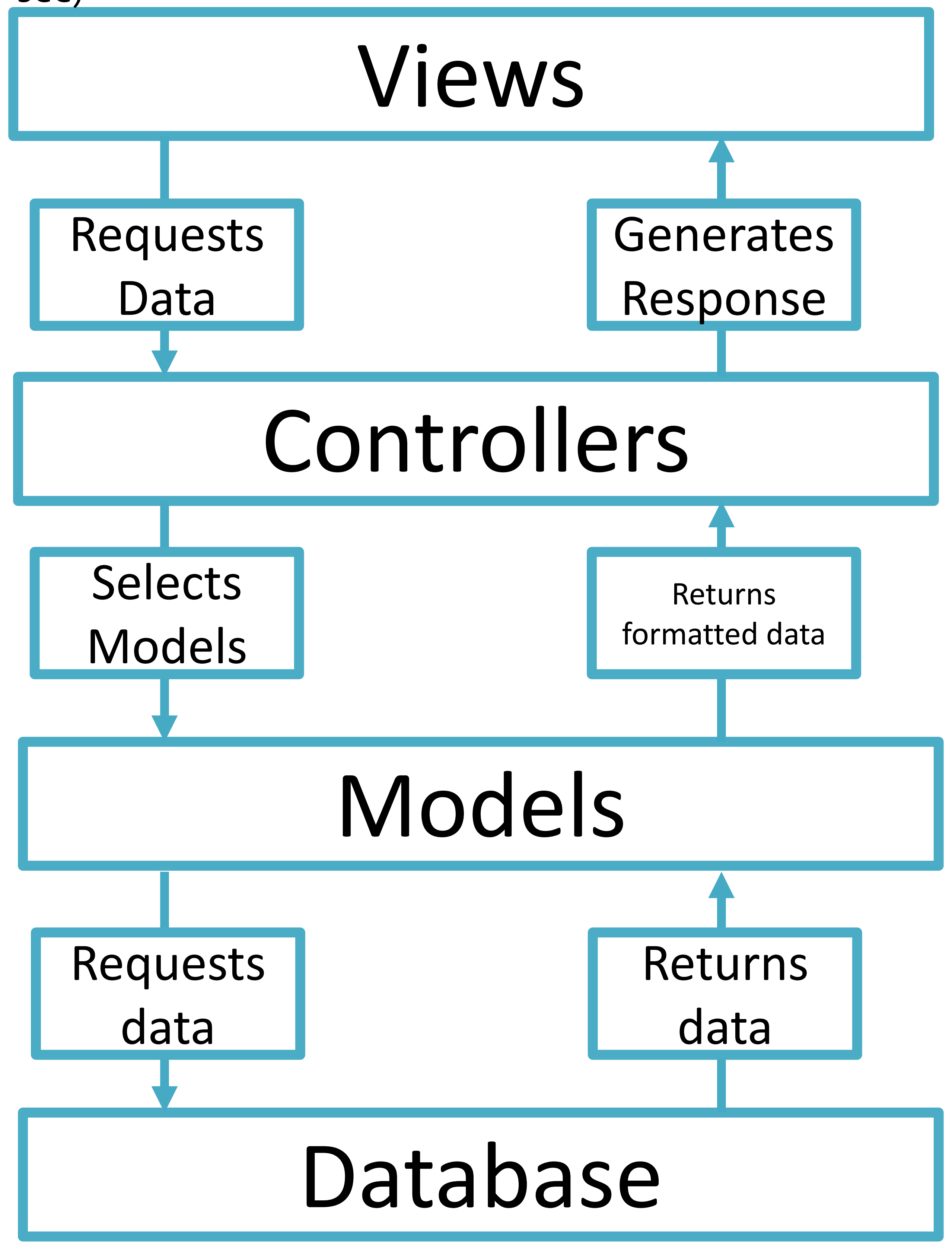
-Detecting relationships between objects
(Partially solved)
-Consistent querying
(Solved in theory. Implimentation in progress)
-Differing conventions
(Partially solved by sticking to REST calls)

Key



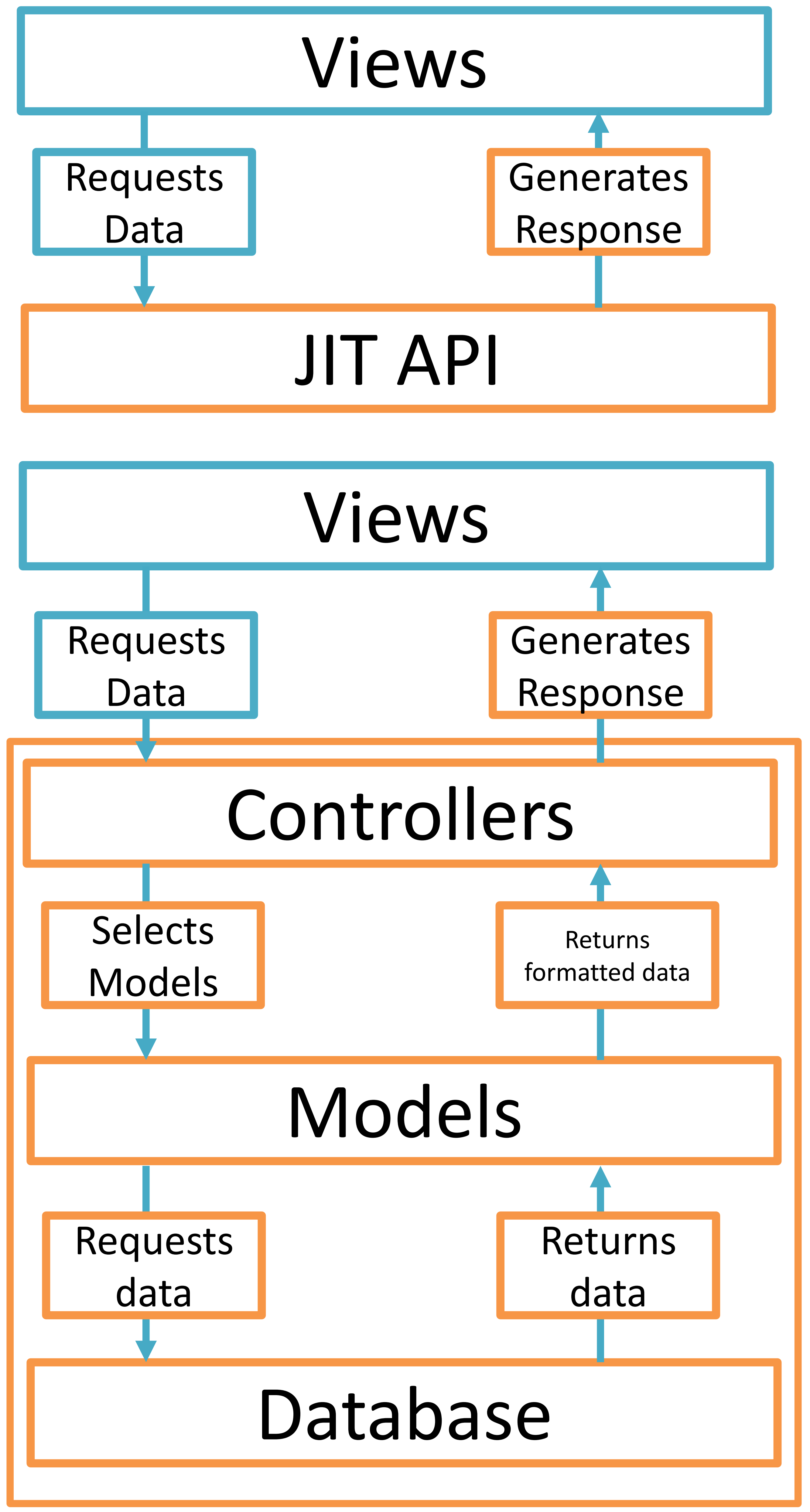
Current Standard

In industry developers work on models, views and controllers which commonly becomes restrictive on views (Generally the most important part of any appliation given that this is what the user and client see)



JIT API alternative

JIT API's propose to replace this style with frontend and test driven code generation (Eliminating the need to write Controllers, Models and SQL)



Alternative Efforts

Other alternatives such as FeathersJS have similar functionality in that they generate REST code from a developer specification. The key difference is they can't generate code from HTTP requests. The use of HTTP requests to generate code means the backend can be more malliable.

Another altenative rest-hapi uses a similar concept to Feathers but suffers similar overhead with developer time for specifying endpoints.

The JIT API solution hopes to reduce the overhead from these other alternatives.

Proof of Concept

LAPI is a work in progress implimentation.
A working MVP with relationships and filtered queries is complete. THE JITAPI and code generation is functional.
Patial solutions have been found to the relationship problem. By enforcing the REST link relation standard we can guarentee correct relationships. In addition to this linking through embedding is possible by using ids that are unique across all models.
The biggest issue with relationships is when an int is used to reference another object. The issue is it could be an int or a relationship and having a program diffentiate between tham is extremly difficult.

Contact Information

▪Github:https://github.com/Dmium/
▪Email: dominiczy.hall@gmail.com
▪LinkedIn: https://www.linkedin.com/in/dominiczyhall/