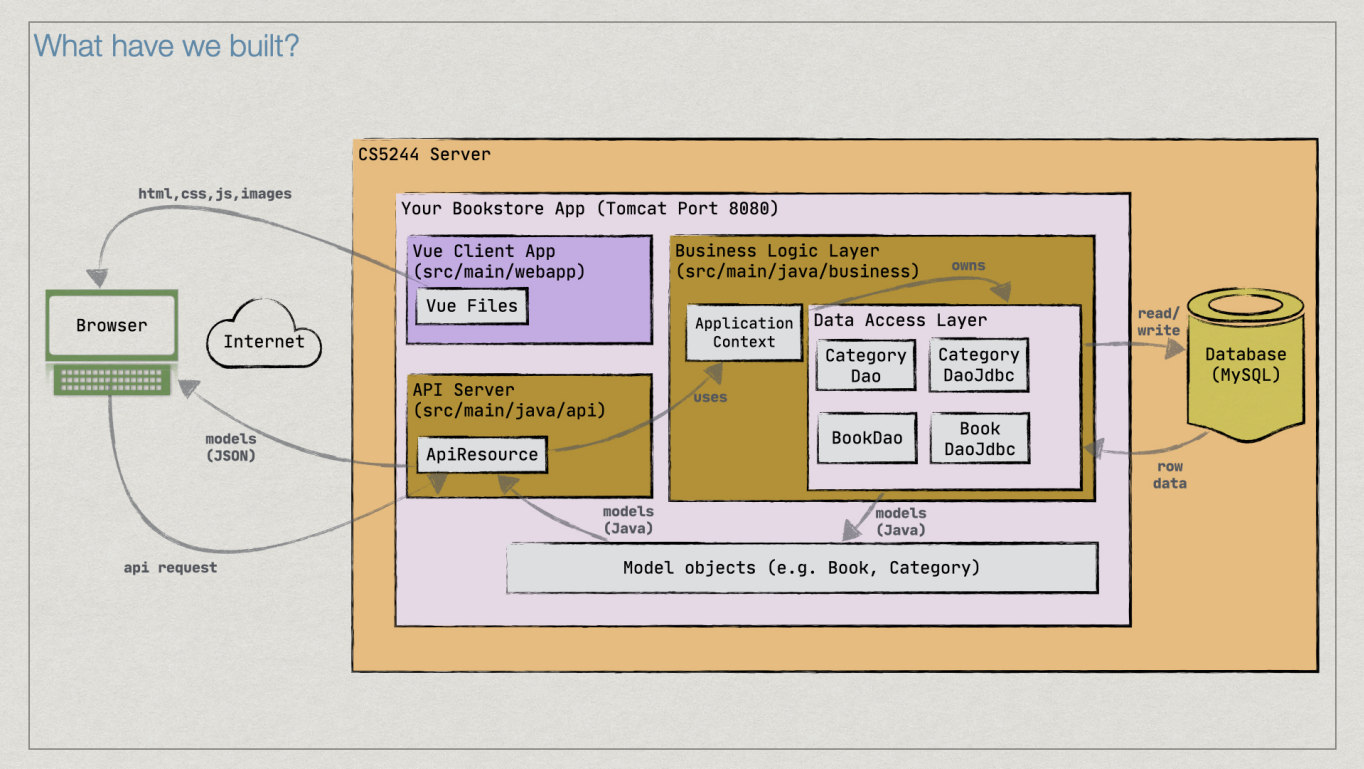
note @280 78 views

A diagram showing what we have built so far...

Folks,

Now that we are combining client and server I wanted to make sure we are not losing sight of the big picture here and to make a couple of additional points about the website architecture and WHY we are building the site this way.

Consider the following diagram for a few minutes - it captures at a high level the structure of the website we are building.



I wanted to make a few points:

- If you have ANY questions as to what is going on in this diagram, please ask. I've made this a question so discussion is encouraged.
- The API layer is separate from the business logic because we could build many different APIs on top of the same business logic. For example, we could build a fulfilment application that tracks orders through different states as the shipping process happens. That is a back-end business operation and is worthy of a second separate API, but they could use the same business logic to read order details for other purposes.
- Also consider how we are using a client project and a server project. This lets different teams work on these projects, and improvements can be made to both at the same time.
- How does this diagram capture the MVC architecture pattern? What is the model, what is the view and what is the controller?

I look forward to any questions here, and I did not want you all to lose sight of the big picture here.

Dr A

#pin

module6 module6/content

Updated 1 month ago by Steven Atkinson

followup discussions *for lingering questions and comments*

○ Resolved ◉ Unresolved

Seth Alexander Brown 1 month ago
Ahh, so Vue is named after "View" in the MVC architecture?
helpful! | 0

Steven Atkinson 1 month ago I'm not 100% sure but it does make sense to me to think of it that way.

Dr A
good comment | 0

Shivam Sharma 1 month ago vue was earlier thought to be named view but there was a already existing package on npm named view so it was named vue.
helpful! | 1

◉ Resolved ○ Unresolved

Joan Piayet Perez Lozano 29 days ago
Hello Dr. A,

I have some questions:

1. If we continue thinking of scalability, apart from having separated the API response from the business logic, would there be other things you would change in this architecture to improve the application? (improve in terms of being an application that could handle high volumes of data or requests)

2. I was also wondering if it is still fine with developing applications with pure java web no frameworks like we are doing here in our project, or do you think there would be a benefit for example if we make a change in the back-end side, switching to Spring framework? or do you think from experience its fine with following the approach we currently have for the back-end?

3. What about changing programming languages? maybe switching to NodeJS or some Python with Django? any benefits here or the approach would be as good as this current one? (so far i think switching programming languages would not make any difference, maybe i am wrong).

4. Maybe to prepare it for a high number of requests we can start thinking of a microservices architecture?

5. For the front-end side, would you have changed something? maybe moving to react js or angular? or no advantages here?

The main intention of my questions is to see how would you have implemented the solution or devised the architecture if this was an app that you were going to launch to real users and without using academic purposes as a criteria to choose the technologies of your stack. I am curious if you would have remained with what we have now, or what technologies would you have changed/added to make it a "highly available" app

Thanks.

helpful! | 0

Steven Atkinson 21 days ago I usually give a talk that explains EXACTLY this question during mid-term week.

I will see if I can give a talk after thanksgiving.

To answer:

1. The basic technology choices here actually run about 200 requests per second given big enough hardware in production for netflix today. The main difference is that our front end servers talk to middle-tier servers that then talk to the database.

2. It's still fine to do things by hand, but only if there's a security win or a performance win.
Many servers use different lighter frameworks or languages. I am partial to using Spring Boot as a technology because it helps with security (packages are vetted) and it make it easier to work nicely with databases. The approach we have uses decent technologies - and it's also important to teach fundamentals here as much as we can without biasing towards a framework - all frameworks in most languages use similar patterns.

3. Mostly it's about building a team with enough experience that forces a language choice. In general node.js is faster to prototype with, and django is very popular so you can find talented folks to work with you there too. People look down on Java as "slow" and "old" now - however it's evolving still, has a great security profile and can use the hardware effectively.

4. I'm a fan of more beefier services than true micro-services. But yes, we need to take pressure off networking basically so splitting up requests into more internal requests works.
Higher request volumes drive the following changes in this order usually:
* move your database to a seperate machine
* beef up the database machine networking and disk speed
* split your api server into customer-data and other-data mid-tier non-micro services
* install a set of healthchecks everywhere and load balancers that take hosts out of rotation when healthchecks fail
* install an outage server to handle the case when everything is overwhelmed

5. Vue shows all the necessary patterns to understand the front end. It's simpler and has better documentation than the others, so a win for teaching.

Technologies:
* load balancers
* web application firewall to block obviously hacky requests
* front, middle and back end services
* big database but then maybe split up databases depending on load.

Thanks for asking such a thorough set of questions.

When I find my slides for that talk I'll share them too.

Dr A

good comment | 0