Full Startup Starter

JVM

Konstantin Ignatyev
kgignatyev@gmail.com

SEATTLE ~~JAVA~~ USER GROUP

SeaJUG

Why we need **FSS** (**F**ull **S**tartup **S**tarter) ?

When I started to search for a new job in a startup type of company (I consider myself the most useful there due to me being ‘specializing generalist’) I got really frustrated: there are some good businesses starting around renewable energy, sustainability, healthcare but they hire for Ruby on Rails and PHP.

Only few startups use JVM ecosystem (Java, Scala, Kotlin), but majority do not even consider it as a possibility.

Quora 15 118 Search Quora

Travis Overby · Follow
Software Engineer at Adobe Systems (company) · 6y

Related **As a beginner, which one should I learn for my startup, Ruby on Rails or PHP with Laravel?**

Igal Alkon · Follow
Building web sites and services since 1998. · Upvoted by Chris Rice, works at Web... · Updated 1y

Related **Should I learn PHP or Ruby? I want to learn just one, due to time constraints. Which one would you choose, and why? If I decide on Ruby will I ever need to use PHP for anything, and vice versa?**

Don’t overthink it, just learn PHP.
It’s easy and fun, and you can build glorious web applications with it!

Andrew Hlavats · Follow
Lives in Toronto, ON · 9y

Related **Should I ditch PHP for Ruby for the sake of startup talent? Would it be worth it to scratch all I've created (the ORM and Back-End) and start learning and coding in RoR just to make it easier for me to find a technical cofounder/developer?**

https://www.quora.com/Should-I-Develop-my-startup-in-PHP-or-Ruby

Ransomware Attackers Quickly Weaponize PHP Vulnerability With 9.8 Severity Rating (arstechnica.com) 6

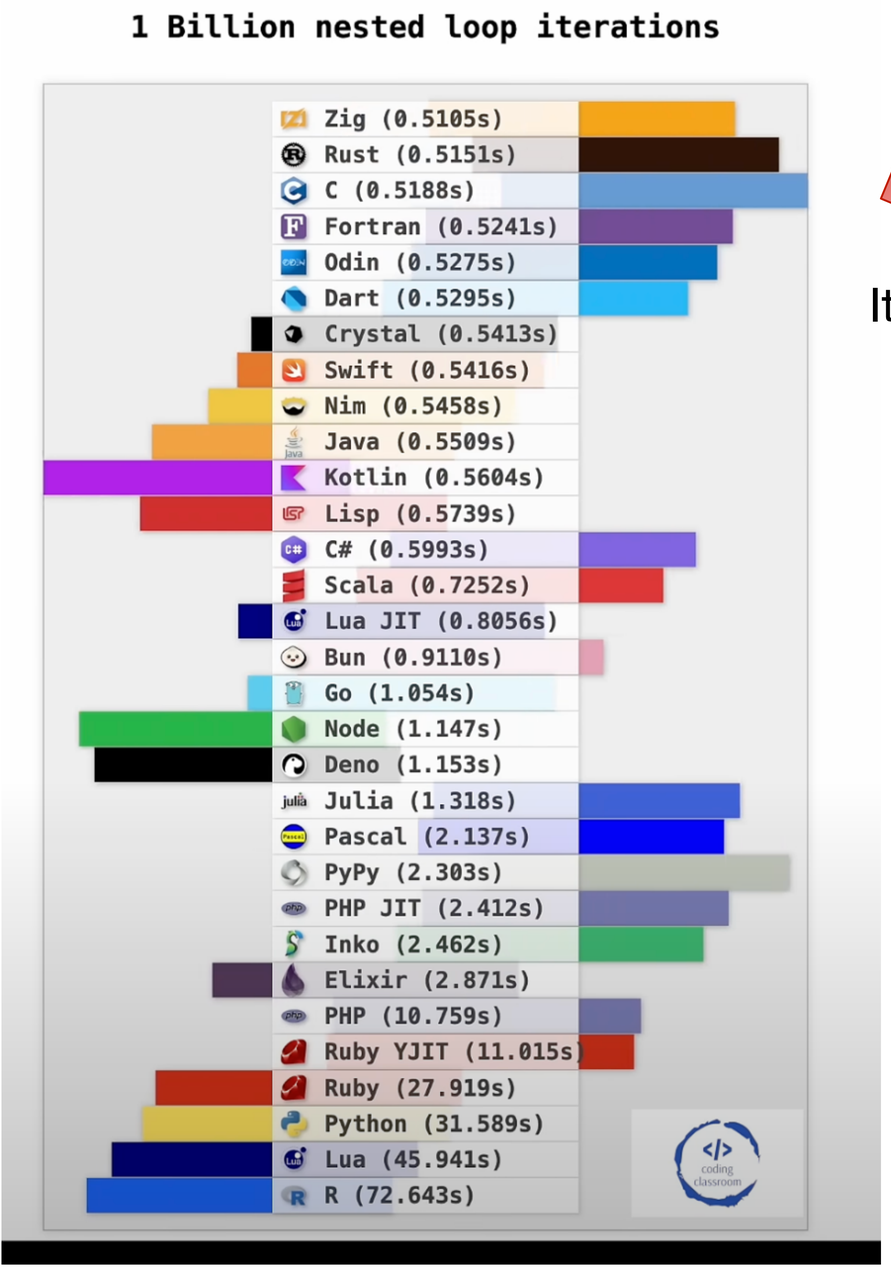
Posted by **BeauHD** on Friday June 14, 2024 @08:45PM from the easy-to-exploit dept.

A critical vulnerability in the PHP programming language (CVE-2024-4577) has been exploited by ransomware criminals, leading to the infection of up to 1,800 servers primarily in China with the TellYouThePass ransomware. This vulnerability, which affects PHP when run in CGI mode, allows attackers to execute malicious code on web servers. Ars Technica's Dan Goodin reports:

⚡ **This is shameful: JVM ecosystem is rich, productive, stable, secure - but only if ‘right’ ingredients are used**

In my career RoR projects had 100% failure rate, and PHP -95% (only because I was out of a company and I didn’t witness its impending collapse)

Why RoR, Python, PHP are considered, and JVM not?



Not because of performance!

Not because of ability to deliver!

It has **Rock-Solid Reliability for Mission-Critical Stuff**

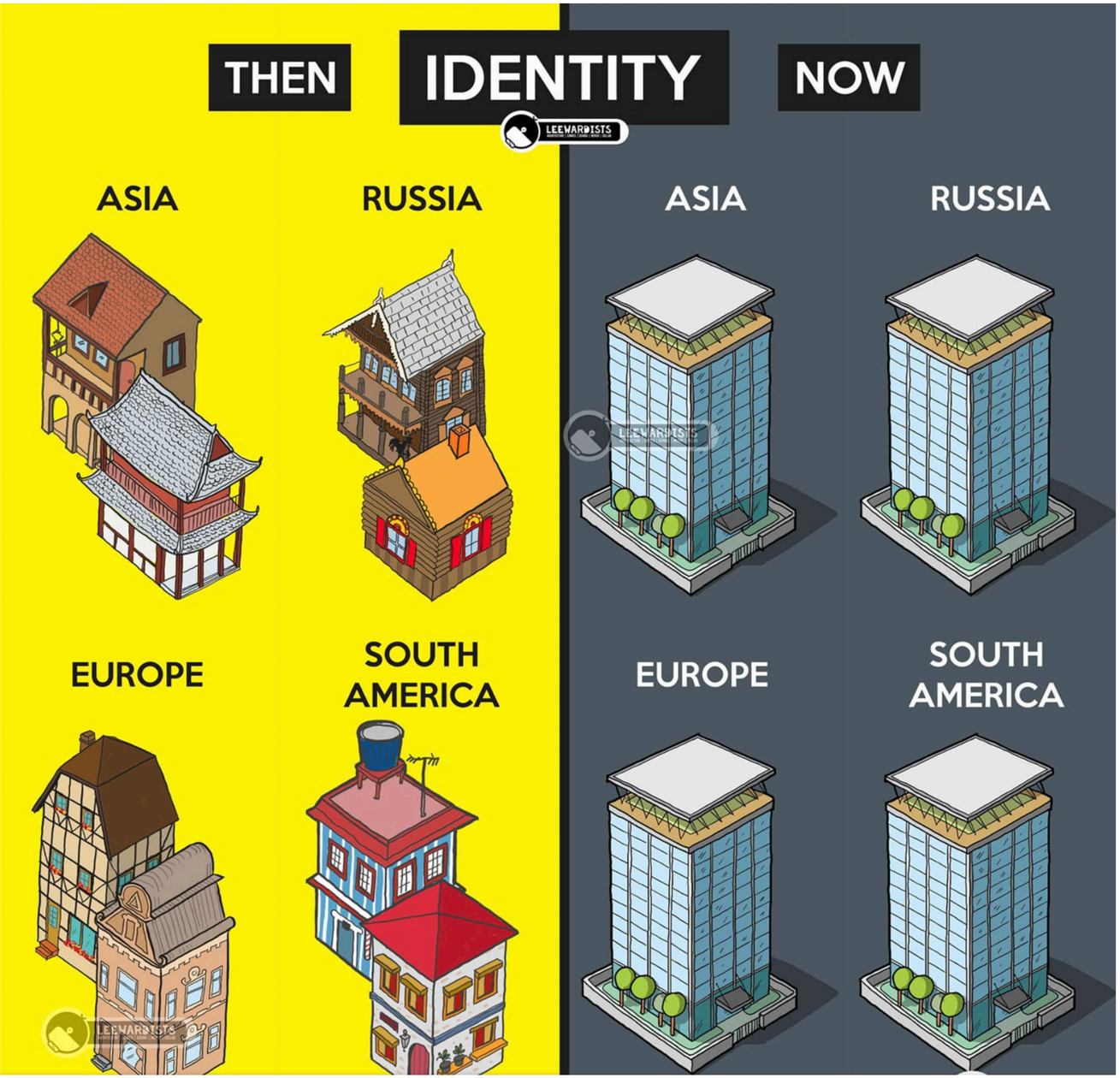
Netflix Still Bets Big on Java:

https://codingplainenglish.medium.com/why-netflix-still-bets-big-on-java-the-strategic-architecture-you-dont-see-afe990cf13f4

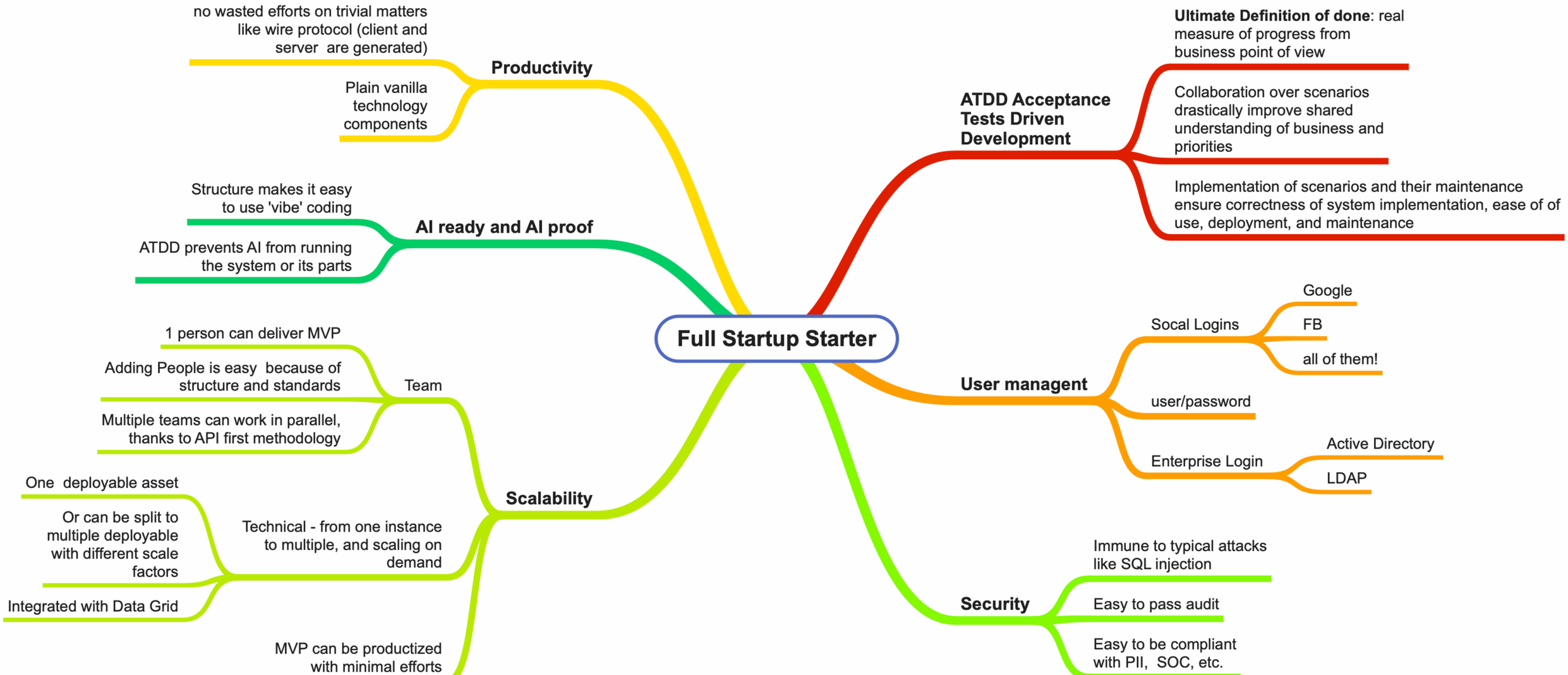
Another more comprehensive test draws similar picture

https://www.techempower.com/benchmarks/#section=data-r23&test=json

Big ‘flattening’ - accelerated by AI

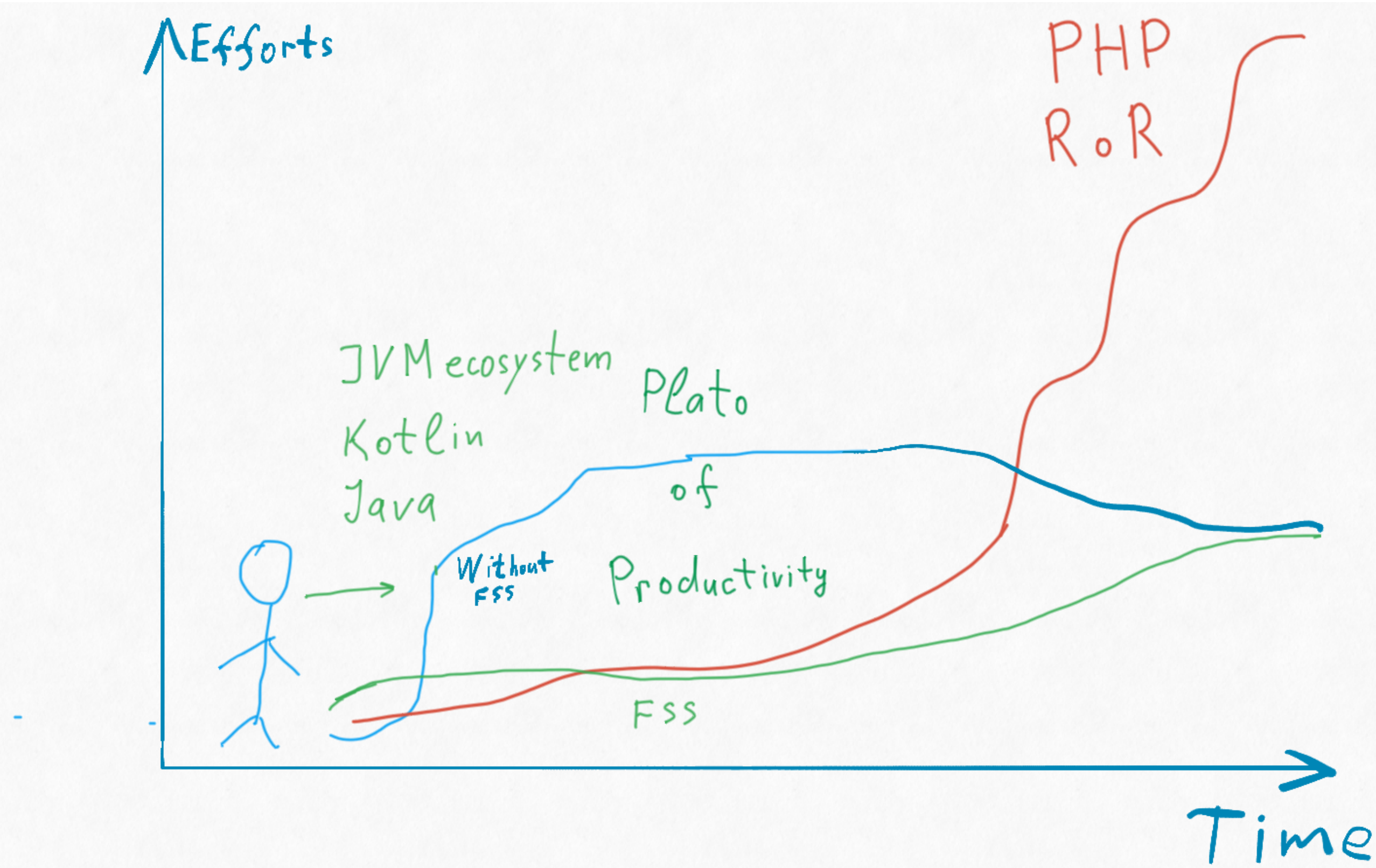


But there is plenty of ‘one-off’-s in ‘pretty’ T-Mobile magenta!


Full Startup Starter
Productivity
no wasted efforts on trivial matters like wire protocol (client and server are generated)
Plain vanilla technology components
AI ready and AI proof
Structure makes it easy to use 'vibe' coding
ATDD prevents AI from running the system or its parts
Scalability
Team
1 person can deliver MVP
Adding People is easy because of structure and standards
Multiple teams can work in parallel, thanks to API first methodology
Technical - from one instance to multiple, and scaling on demand
One deployable asset
Or can be split to multiple deployable with different scale factors
Integrated with Data Grid
MVP can be productized with minimal efforts
ATDD Acceptance Tests Driven Development
Ultimate Definition of done: real measure of progress from business point of view
Collaboration over scenarios drastically improve shared understanding of business and priorities
Implementation of scenarios and their maintenance ensure correctness of system implementation, ease of of use, deployment, and maintenance
User managent
Socal Logins
Google
FB
all of them!
user/password
Enterprise Login
Active Directory
LDAP
Security
Immune to typical attacks like SQL injection
Easy to pass audit
Easy to be compliant with PII, SOC, etc.

Why we need **FSS** (**F**ull **S**tartup **S**tarter) ?

It should make realization of JVM ecosystem benefits much easier for startups and projects of any size and complexity. Making sure anyone can start easy and scale as much as they need, when they need.



FSS Principles - opinionated setup

Assumptions about your project:

- it is more than a CRUD UI on top of a DB;
- it should be easy to create MVP;
- it should be easy to scale team;
- it should be easy to implement complex logic and integrations;
- it should be easy to pass compliance with SOC, HIPPA, PII, etc;
- it should be easy to scale performance and throughput;

Principles:

- Acceptance Tests Driven Development
  * ultimate definition of done and amazing productivity boost
- Interface First Design and Development
- Plain vanilla technology components without 'wrappers'
  * swappable if desired, for example change JPA to myBatis or JOOQL
- Typesafe languages
- Transferrable and reusable knowledge
- Scalable structure
  * start with 'modularity' and architecture controls
  * complex interactions delegated to Temporal.io
  * modules can be turned into separate services if needed
- Beginner and 'Vibe coding' friendly
- 'Vibe coding' safe

Starter:

A fully functional service with a UI that has batteries included:
- easy to add new features;
- integrated with Single Sign On;
- API-s are exposed and ready to serve any client;
- robust Authorization Model;
- supports database management and evolution;
- can be easily refactored with IDE and search/replace tools to bootstrap a startup or a project of any complexity;

Why not X, Y, or Z - JHipster

JHipster Domain Language (JDL)

Improve this page

The JDL is a JHipster-specific domain language where you can describe all your applications, deployments, entities and their relationships in a single file (or more than one) with a user-friendly syntax.

You can use our online JDL-Studio or one of the JHipster IDE plugins/extensions, which are available for:

- IntelliJ
- Eclipse,
- VS Code

Upgrading an application

Improve t

Summary

1. Option 1 - automatic upgrade
2. Option 2 - manual upgrade

Option 1 - automatic upgrade

When a new version of JHipster is released, the JHipster upgrade sub-generator helps upgrade an existing application to this new version, without erasing your changes.

Creating an entity

Introduction

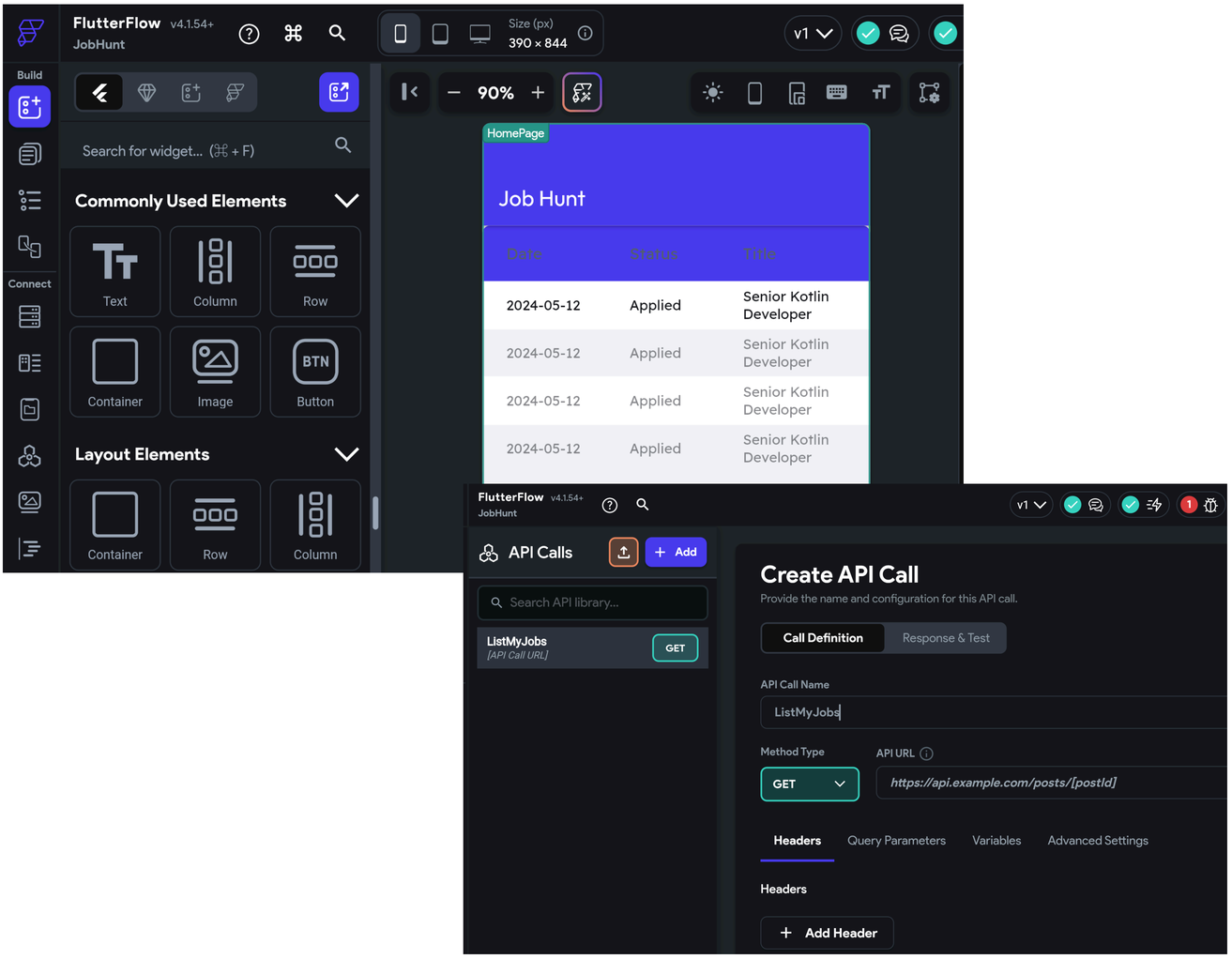
Once you have created your application, you will want to create *entities*. For example, *Author* and a *Book* entity. For each entity, you will need:

- A database table
- A Liquibase change set
- A JPA Entity
- A Spring Data JPA Repository
- A Spring MVC REST Controller, which has the basic CRUD operations
- An Angular router, a component and a service
- An HTML view
- Integration tests, to validate everything works as expected
- Performance tests, to see if everything works smoothly

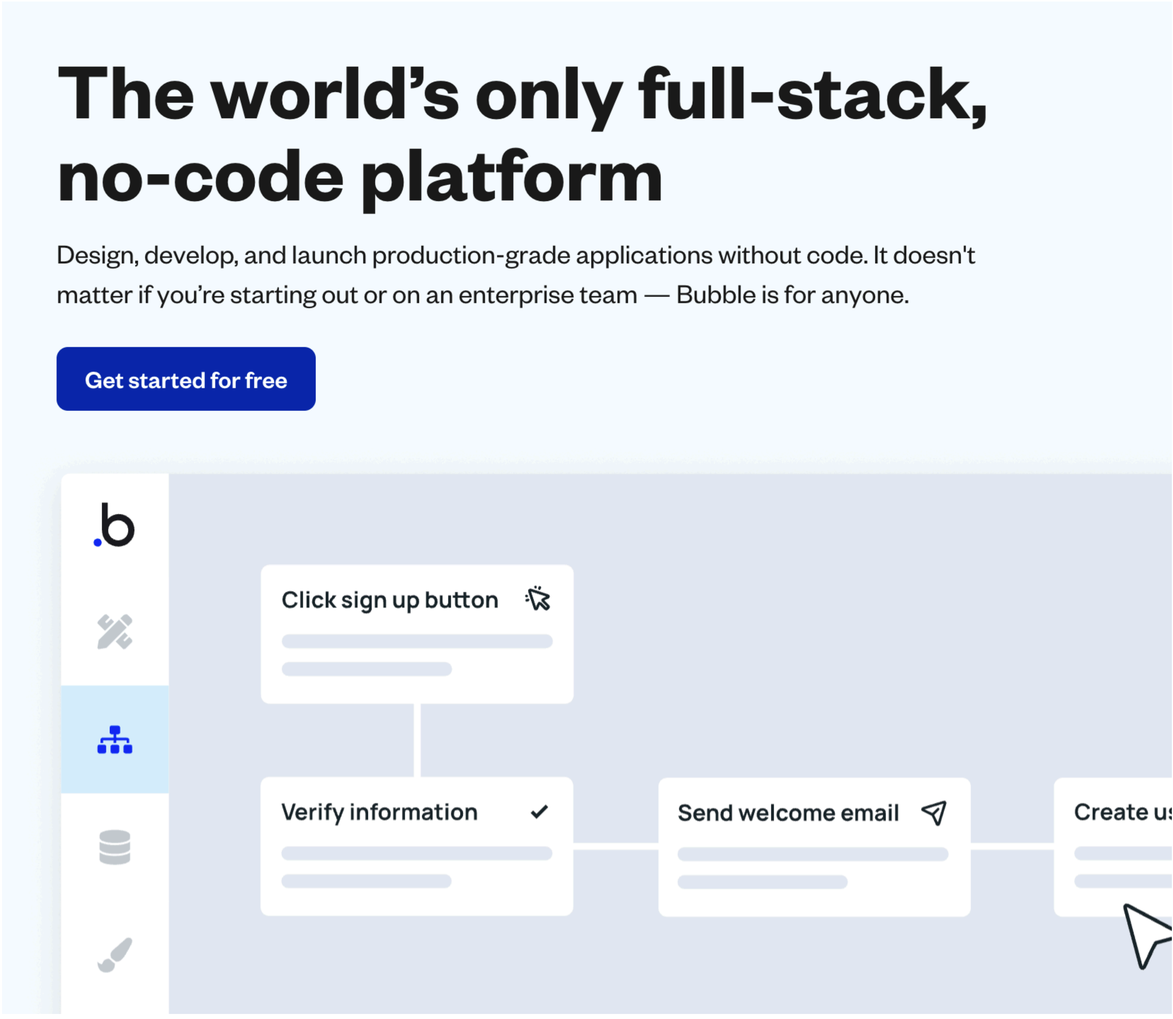
In a few words:
- it tries to do too much for folks who are just starting - ‘JHipster’ magic on top of technologies in use - it is built around idea of ‘metaprogramming’;
- it is just another way to make UI on top of a database – I think such applications and services are things of the past. If a DB needs a UI then DB can be ‘automagically’ exposed as via GraphQL plus a visual UI builder like FlutterFlow.io or Bubble.io

Why not X, Y, or Z - FlutterFlow and Bubble etc

https://flutterflow.io/



https://bubble.io/



Visual form editors exist for a looooooong time, Oracle Form, Delphi, and many others come to mind. Not we have them online and they work with API-s. While they have their utility I doubt that they are revolutionary – they have their place but for anything decent we need full control over code and debuggability. So far nothing works better than a good type-safe language and an IDE.

https://medium.com/@CodeOps/enterprise-java-isnt-dead-it-s-just-smiling-through-the-pain-1788a52d2b8e

Enterprise Java Isn't Dead, It's Just Smiling Through the Pain

Spring Boot, microservices, and slow death by annotation.

```
@RestController
@RequestMapping("/api/v1/users")
@CrossOrigin(origins = "http://localhost:3000")
@Validated
@Slf4j
public class UserController {
    @Autowired
    private UserService userService;

    @GetMapping("/{id}")
    @PreAuthorize("hasRole('USER')")
    @Cacheable(value = "users", key = "#id")
    @ApiOperation(value = "Get user by ID", response = UserDto.class)
    public ResponseEntity<UserDto> getUserById(
            @PathVariable @Valid @NotNull Long id,
            @RequestHeader(value = "X-Correlation-ID", required = false) String

        log.info("Fetching user with ID: {}, correlation: {}", id, correlationId
        UserDto user = userService.findById(id);
        return ResponseEntity.ok(user);
    }
```

Let's discuss what problems we can find in this code

- Violation of 12 factor principles - config should be external;
- Authorization should be inside service(s);
- API specification is generated from code (@ApiOperation);
- Caching should be applied by service;

Http and Rest controller is just one way the business code can be called, messages and workflow actions are not going to use this class.

https://medium.com/@CodeOps/enterprise-java-isnt-dead-it-s-just-smiling-through-the-pain-1788a52d2b8e

Enterprise Java Isn't Dead, It's Just Smiling Through the Pain

Spring Boot, microservices, and slow death by annotation.

```
@EnableEurekaClient
@EnableCircuitBreaker
@EnableZuulProxy
@SpringBootApplication
public class OrderServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(OrderServiceApplication.class, args);
    }
}
@RestController
@RequestMapping("/orders")
public class OrderController {
    @Autowired
    private OrderService orderService;
    @Autowired
    private PaymentServiceClient paymentClient;
    @Autowired
    private InventoryServiceClient inventoryClient;
    @CircuitBreaker(name = "orderService", fallbackMethod = "createOrderFallback")
    @Transactional
    public ResponseEntity<Order> createOrder(@RequestBody OrderRequest request) {
        // Orchestrate multiple service calls
        PaymentResponse payment = paymentClient.processPayment(request.getPaymentI
        InventoryResponse inventory = inventoryClient.reserveItems(request.getItem

        if (payment.isSuccess() && inventory.isSuccess()) {
            Order order = orderService.createOrder(request, payment.getId(), inver
            return ResponseEntity.ok(order);
        }

        throw new OrderCreationException("Failed to create order");
    }
    public ResponseEntity<Order> createOrderFallback(OrderRequest request, Excepti
        log.error("Order creation failed, falling back", ex);
        return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE)
                .body(new Order("FAILED", "Service temporarily unavailable"));
    }
}
```

This looks clean until you realize that you need:

- Service discovery (still Eureka, despite being legacy)
- Circuit breakers (now Resilience4J since Hystrix is deprecated)
- Load balancing (Spring Cloud LoadBalancer replaced Ribbon)
- Configuration management (Spring Cloud Config)
- Distributed tracing (Micrometer Tracing)
- API Gateway (Spring Cloud Gateway)

Each of these introduces latency, complexity, and potential failure points. We're paying a performance tax for architectural complexity that may not be worth it.

https://medium.com/@CodeOps/enterprise-java-isnt-dead-it-s-just-smiling-through-the-pain-1788a52d2b8e

Enterprise Java Isn't Dead, It's Just Smiling Through the Pain

Spring Boot, microservices, and slow death by annotation.

```
@Entity
@Table(name = "products")
@EntityListeners(AuditingEntityListener.class)
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@EqualsAndHashCode(callSuper = false)
@ToString(exclude = {"reviews", "categories"})
@Indexed // For search functionality
@EventListeners(ProductEventListener.class)
public class Product extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false, length = 100)
    @NotBlank(message = "Product name is required")
    @Size(min = 1, max = 100, message = "Product name must be between 1 and 100
    @Field(type = FieldType.Text, analyzer = "standard") // Elasticsearch
    private String name;
    @Column(columnDefinition = "TEXT")
    @NotBlank(message = "Product description is required")
    @Size(min = 10, max = 1000, message = "Description must be between 10 and 10
    @Field(type = FieldType.Text, analyzer = "standard")
    private String description;
    @Column(nullable = false, precision = 10, scale = 2)
    @DecimalMin(value = "0.01", message = "Price must be greater than 0")
    @DecimalMax(value = "999999.99", message = "Price cannot exceed 999,999.99")
    @JsonSerialize(using = MoneySerializer.class)
    private BigDecimal price;
```

Let's discuss what problems we can find in this code

How ‘investing’ in FSS benefits developers and businesses - Part 1. Language

	Go	Ruby	PHP	Java	Kotlin
Can write cross platform services	✓	✓	✓	✓	✓
Performance	Fast	Slowest	Slow	Fastest	Fastest
Typesafe	✓	✗	✗	✓	✓
Can be used to build mobile applications	✗	✗	✗	✗✓ Android only	✓
Can be used in Jupiter Notebook	maybe	✓	maybe	✓	✓
Co-routines	✓	✗	✗ kind of	✗✓ virtual threads	✓
Compile to native code directly	✓	✗	✗	✗	✓
Pleasant modern language	✗	✗	✗	✗✓ trying to catch up	✓

https://www.techempower.com/benchmarks/#section=data-r23&test=update

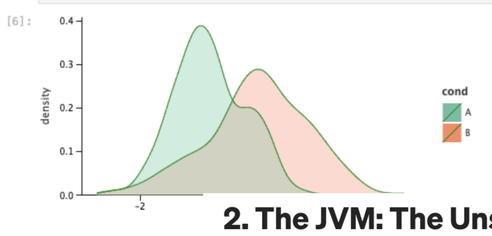
Kotlin Multiplatform

NETFLIX PHILIPS fastwork AUTODESK shopify PowerSchool vmware Meetup POLARSTEPS Careem LEROY MERLIN Cash App deezer Footballco teamSNAP

Kotlin

JVM JS Native

```
[6]: // Sample taken from https://github.com/JetBrains/lets-plot-kotlin
val rand = java.util.Random()
val data = mapOf<String, Any>(
    "rating" to List(200) { rand.nextGaussian() } + List(200) { rand.next(
    "cond" to List(200) { "A" } + List(200) { "B" }
)

var p = lets_plot(data)
p += geom_density(color="dark_green", alpha=.3) {x="rating"; fill="cond"}
p + ggsize(500, 250)
```



2. The JVM: The Unsung Hero That Makes It All Fly:

The Java Virtual Machine (JVM) is, hands down, the MVP here. It’s not just a place where Java code runs; it’s an insanely optimized beast. Modern JVMs have these things called “Just-In-Time” (JIT) compilers.

https://codingplainenglish.medium.com/why-netflix-still-bets-big-on-java-the-strategic-architecture-you-dont-see-afe990cf13f4

So, Why Java for Netflix? Let’s Talk Engineering

1. Rock-Solid Reliability for Mission-Critical Stuff:

Just change compiler and start writing more robust code

```
kotlin-misc [kotlin-misc-tests]  ~/dev/k
  .idea
  src
    main
      kotlin
        kgi.test
          Main.kt
          SomeJavaClass
          TestDC.kt
      resources
    test
  target
  .gitignore
  pom.xml
External Libraries
Scratches and Consoles
```

SomeJavaClass.java TestDC.kt

```
2   <project xmlns="http://maven.apache.org/POM/4.0.0"
23
24      <build>
25          <sourceDirectory>src/main/kotlin</sourceDirectory>
26          <testSourceDirectory>src/test/kotlin</testSourceDirectory>
27          <plugins>
28              <plugin>
29                  <groupId>org.jetbrains.kotlin</groupId>
30                  <artifactId>kotlin-maven-plugin</artifactId>
31                  <version>2.0.0</version>
32                  <executions>
33                      <execution>
34                          <id>compile</id>
35                          <phase>compile</phase>
36                          <goals>
37                              <goal>compile</goal>
38                          </goals>
39                      </execution>
40                      <execution>
41                          <id>test-compile</id>
42                          <phase>test-compile</phase>
43                          <goals>
44                              <goal>test-compile</goal>
45                          </goals>
46                      </execution>
47                  </executions>
```

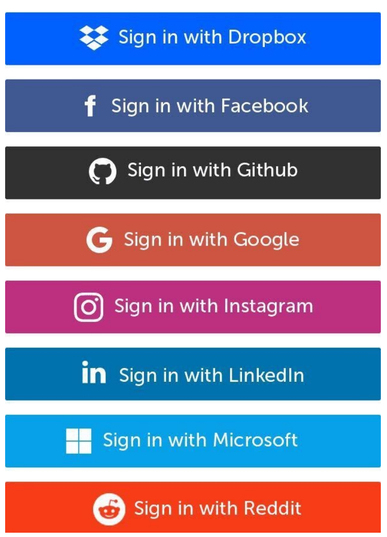
```
16  fun main() {  new *
17      val rnd = SomeJavaClass().randomString
18      println(rnd)
19
20      val fName = findPersonRecord( id
21      println("Hello ${fName}!")
22  }
```

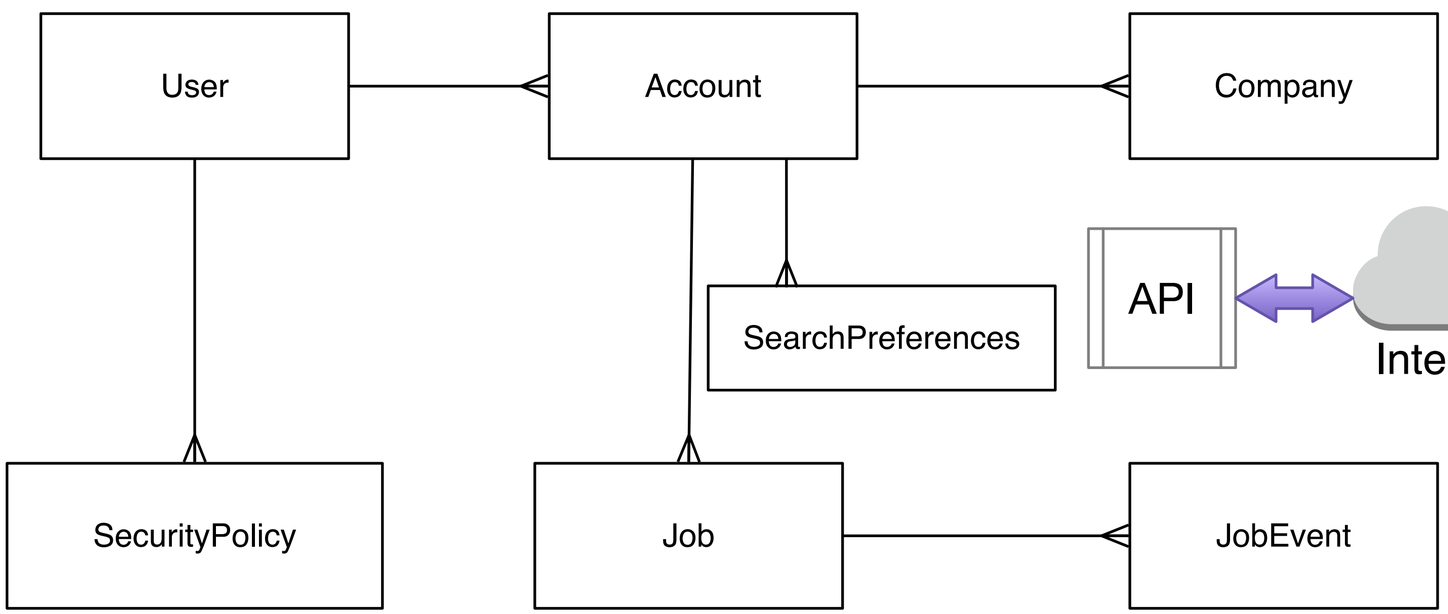
kgi.test.SomeJavaClass
public String getRandomString()
kotlin-misc-tests

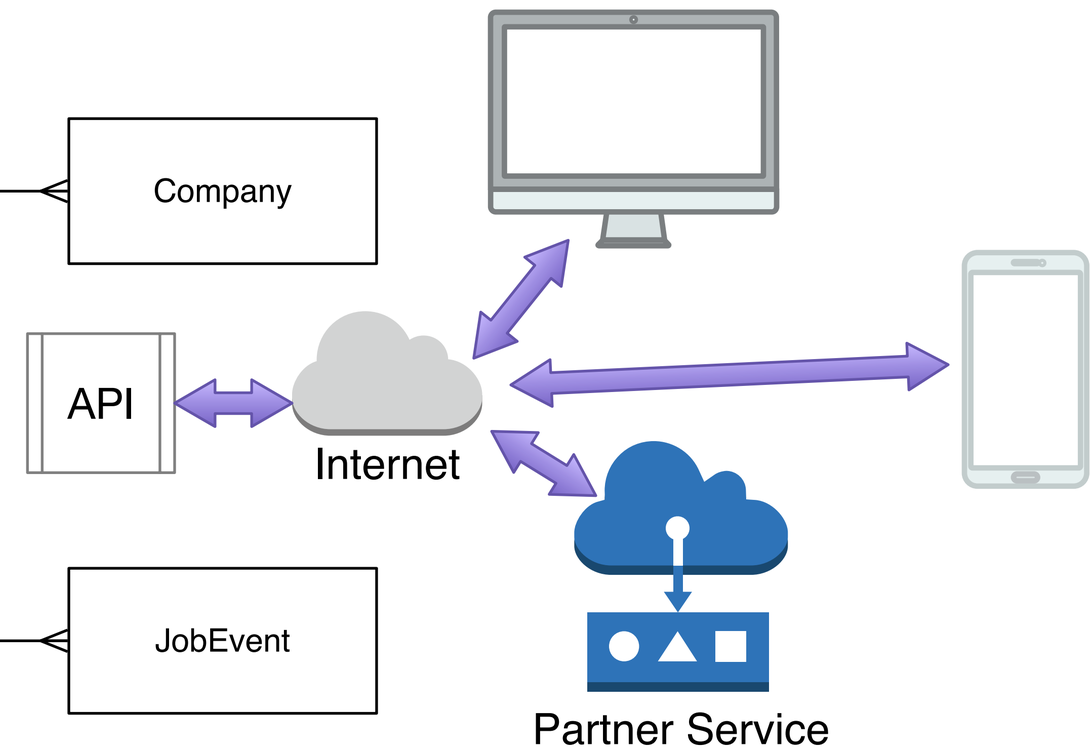
FSS Principles - opinionated setup

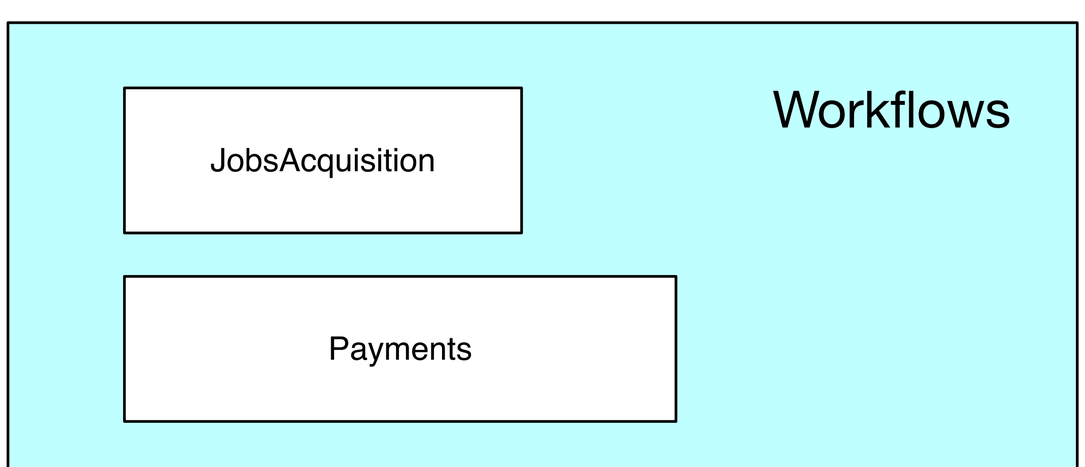
Let's pretend that we will be building a true Job Seeker service to allow people track their application, auto detect duplicates etc.

And we want to allow mobile clients, 3-rd party plugins, partners, SSO, Social Logins, observability, compliance, fault tolerance, etc.



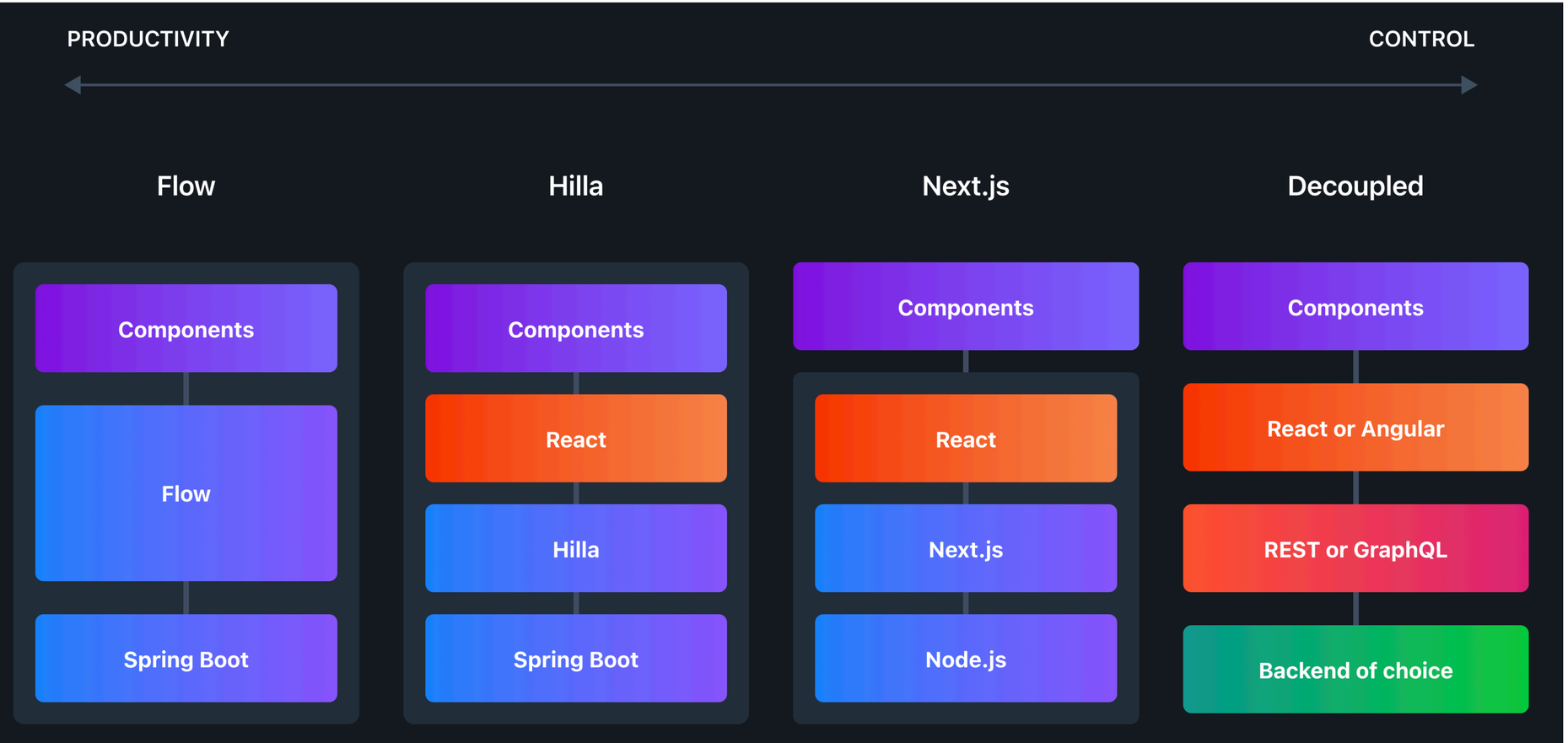


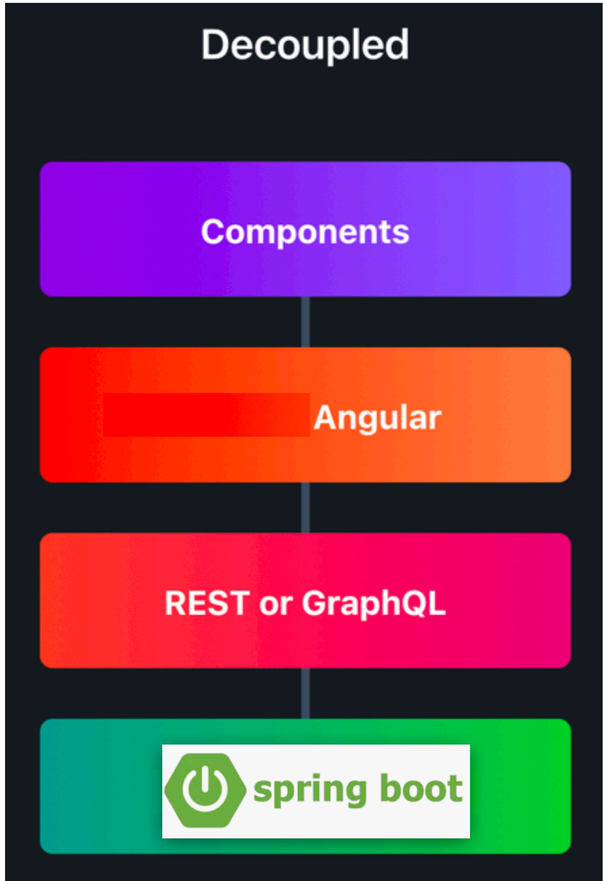




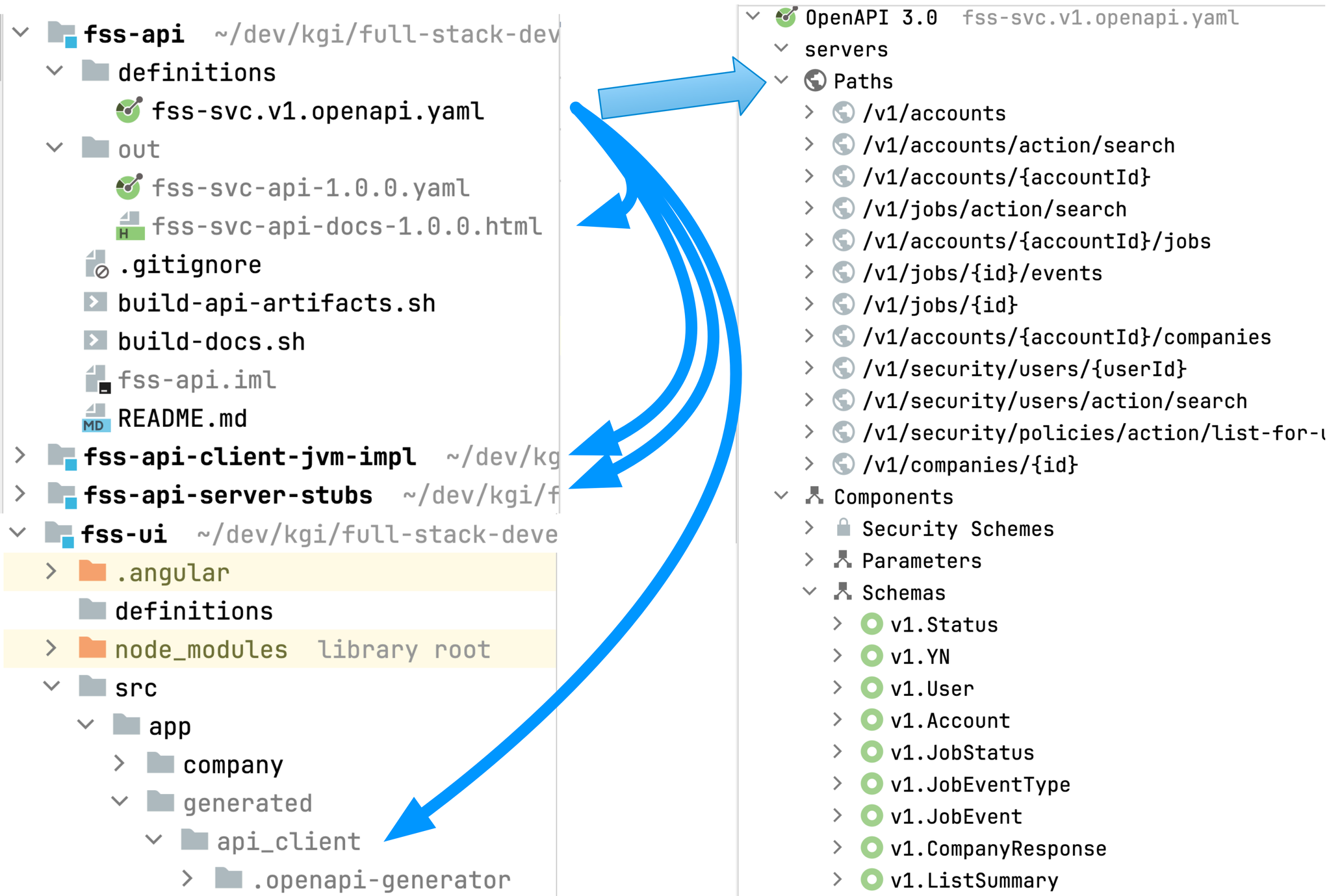
Folks from Vaadin claim that decoupled is less productive - not true!

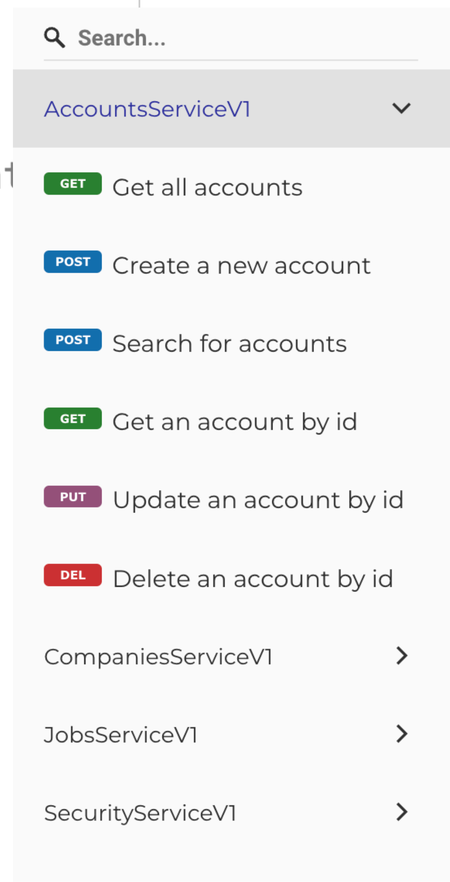
https://vaadin.com/

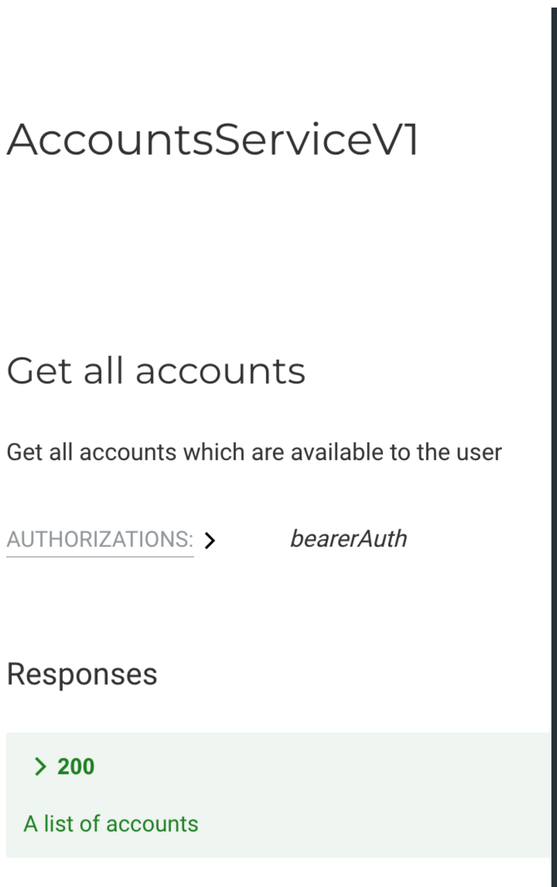


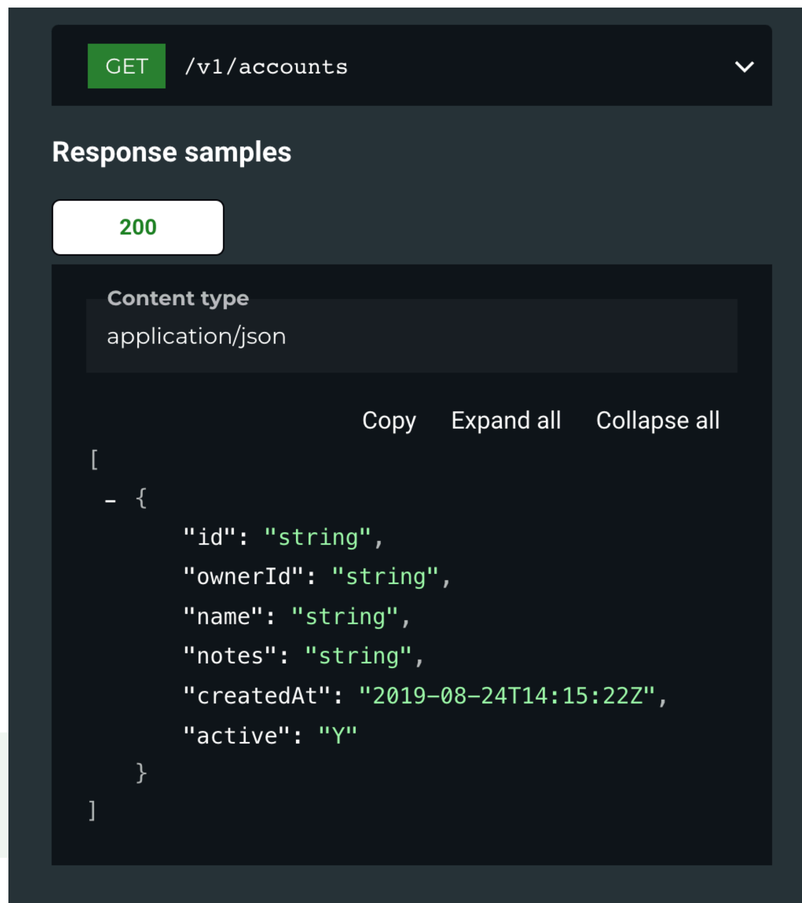


API- First, define once and generate clients, server, and docs



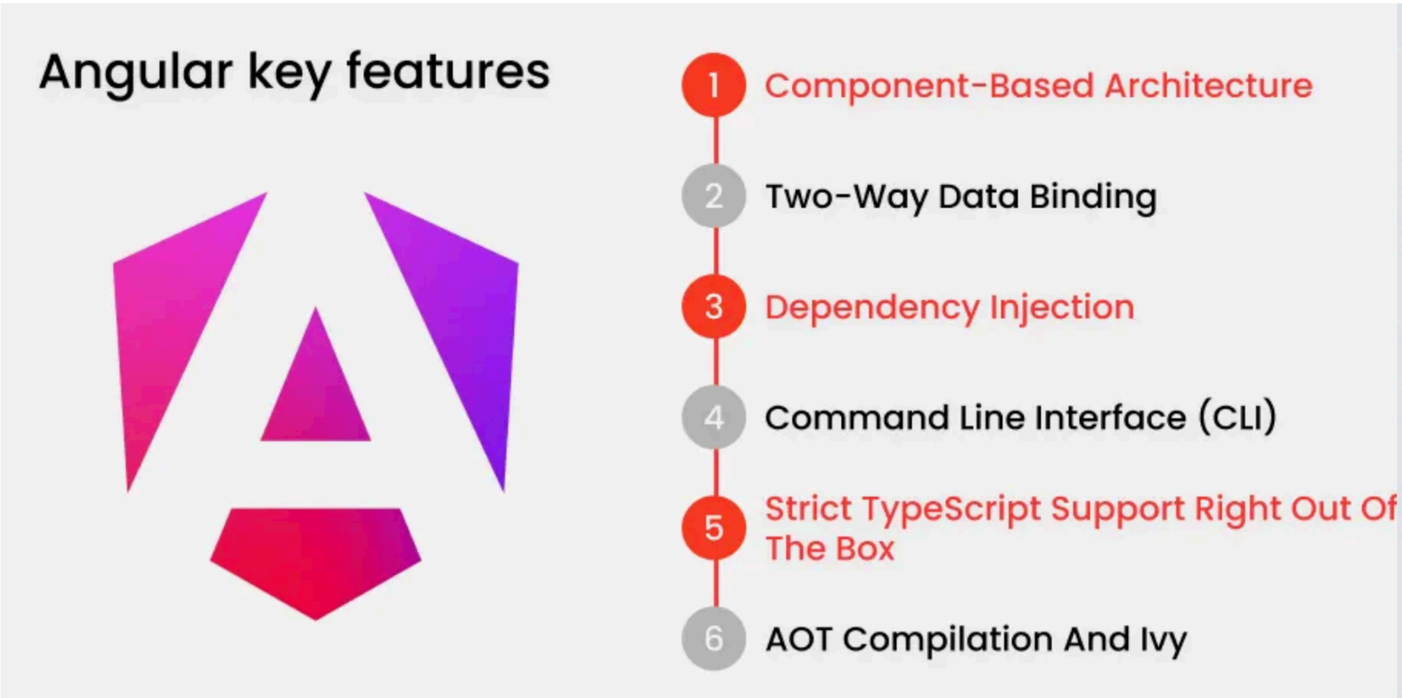


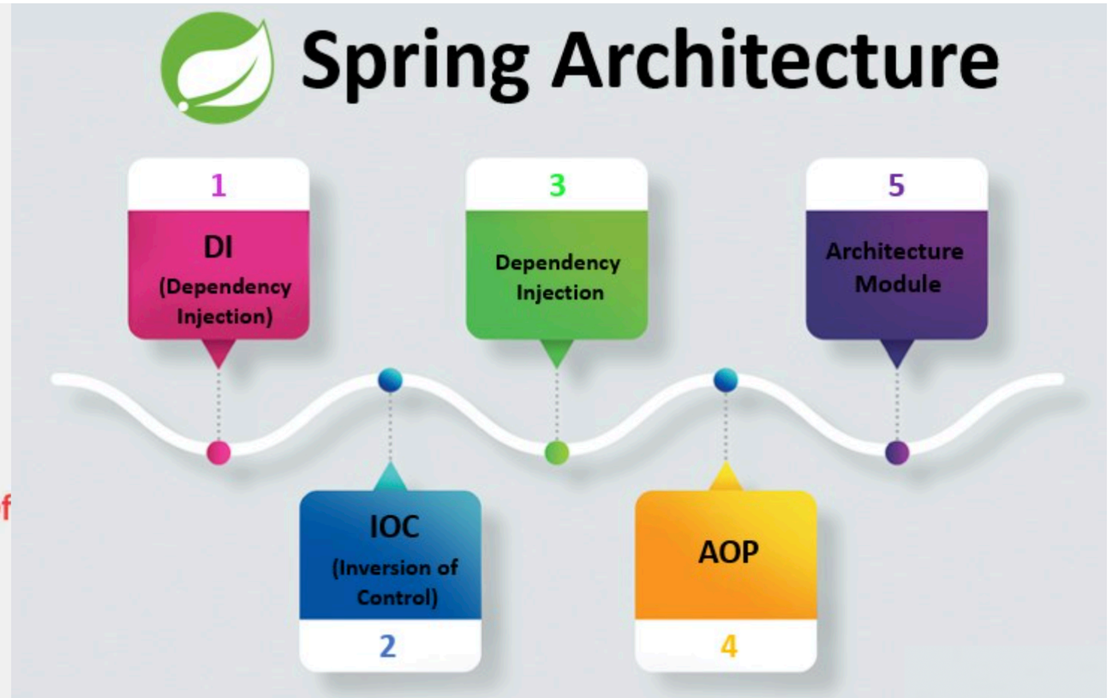




How ‘investing’ in this technology set benefits developers (and businesses) - Part 2. Frameworks

If you know Spring framework - you can become productive with Angular in a half of a day





You saw one Angular project: you saw them all! - very consistent and easy to navigate and reason about.

You saw one React project - you only know one React project

Spring

```
@Service  kgignatyev *
@Transactional
open class CompaniesSvcImpl(
    val _companiesRepo: CompaniesRepo,
    val accountsSvc: AccountsSvc,
    val securitySvc: SecuritySvc
): CompaniesService, CrudRepository<Company, String>
```

Angular

```
@Component({  Show usages  kgignatyev *
    selector: 'app-companies-list',
    templateUrl: './companies-list.component.html',
    standalone: false
})
export class CompaniesListComponent {

  companies: Array<V1Company> = [];
  pageSize: number = 15;

  constructor(private companiesSvc: CompaniesServiceV1Service,
              private cxtSvc:ContextService,
              private router: Router) {
```

```
@RequestMapping(  1 override
        method = {RequestMethod.POST},
        value = {"/v1/accounts/{accountId}/companies"},
        produces = {"application/json"},
        consumes = {"application/json"}
)
default ResponseEntity<V1Company> createCompany(@PathVariable("accountId")
```

```
@RequestMapping(
        method = {RequestMethod.DELETE},
        value = {"/v1/companies/{id}"},
        produces = {"application/json"}
)
default ResponseEntity<V1Status> deleteCompanyById(@PathVariable("id") String id)
```

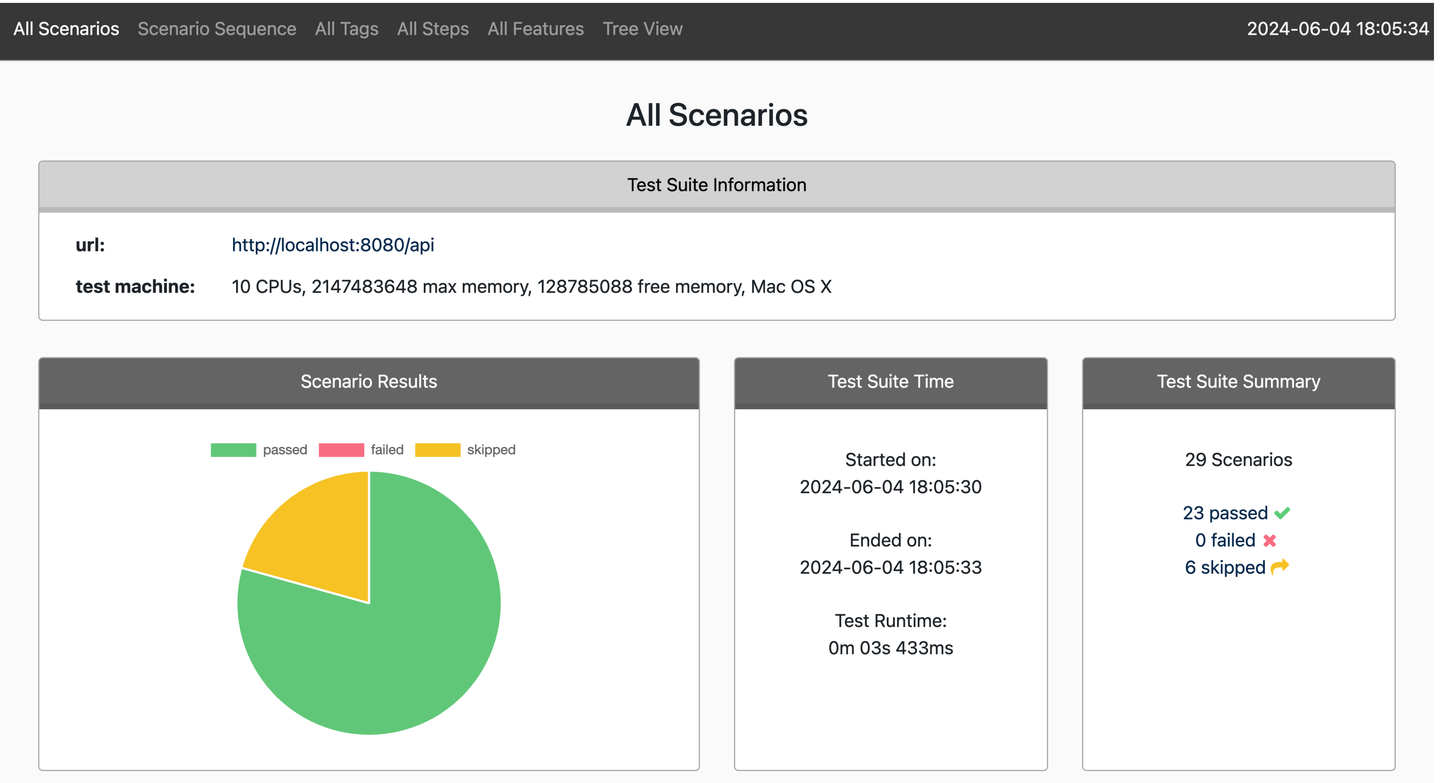
```
const routes: Routes = [

  {path: 'companies', component: CompaniesListComponent},
  {path: 'companies/:id', component: CompanyComponent},
  {path: 'jobs', component: JobsListComponent},
  {path: 'jobs/:id', component: JobComponent},
  {path: '', component: AboutComponent},
]
```

ATDD - the starting point

```
fss-acceptance-test [fss-service-acceptance
> .secrets
> config
v src
  v main
    > kotlin
    v resources
      v scenarios
          1init.feature
          account-management.feature
          jobs-management.feature
          scalability.feature
          security.feature
          system-management.feature
```

```
14  Scenario Outline: Update job
15    Given user "<userName>" is logged in
16    And current user can find "<jobName>"
17    Then current user can update current job notes
18
19    Examples:
20      | userName  | jobName    |
21      | testUser1 | TestJob1-1 |
22      | testUser2 | TestJob1-2 |
23
24  Scenario Outline: Record job history
25    Given user "testUser1" is logged in
26    And current user can find "TestJob1-1"
27    Then current user can add event "<jobEventType>" happened on "<date>" with notes "<notes>"
28    And check that job has status: "<jobStatus>"
29
30    Examples:
31      | jobEventType        | date       | notes                              | jobStatus           |
32      | APPLIED             | 2024-05-01 | Applied for the job                | APPLIED             |
33      | INTERVIEW_SCHEDULED | 2024-05-01 | initial call with recruiter        | INTERVIEWING        |
34      | INTERVIEWED         | 2024-05-01 | Technical interview with John Dow  | INTERVIEWING        |
35      | REJECTED            | 2024-05-20 | Rejection note                     | REJECTED_BY_COMPANY |
```

ATDD - benefits



- it literally bring business folks and developers on ‘the same page’;
- it establishes common language and terminology;
- it shows true business progress - not superficial burn-down charts - team can have them, but they do not matter;
- makes prioritization and focusing easy;
- allows evaluation of API-s and assumptions before a single line of implementation code is written;
- serves as ultimate definition of ‘done’;
- makes project ‘Vibe code’ safe;

https://medium.com/@kgignatyev/acceptance-tests-driven-development-atdd-b3a6420235f4

ATDD - early warnings system

2 FAILED	4 PASSED	14 PENDING	2 UNDEFINED

18% passed 22 executed	**10 seconds ago** last run	**2.8 seconds** duration
Mac OS X	OpenJDK 64-Bit Server VM 21.0.3+9-LTS	cucumber-jvm 7.15.0

Search with text or @tags

You can search with plain text or Cucumber Tag Expressions to filter the output

failed passed pending undefined

classpath:scenarios/1init.feature

classpath:scenarios/account-management.feature

@test-account

Feature: Account management

Back to outline and all 2 examples

@test-account

Example: Create a new account

- **Given** user "testUser1" is logged in
- **And** current user does not have an account
- **And** current user is not stored in the system
- **Then** current user can create a new account with the name "Test1"

```
com.kgignatyev.fss_svc.api.fss_client.v1.infrastructure.ServerException: Server error : 500  org.springframework.dao.DataIntegrityViolationException: could not execute statement [ERROR: insert or update on table "accounts_acnt" violates foreign key constraint "accounts_acnt_owner_id_fkey"
  Detail: Key (owner_id)=() is not present in table "users_usrs".] [insert into jobs.accounts_acnt (active,created_at,email,name,owner_id,updated_at,acnt_id) values (?,?,?,?,?,?,?)]; SQL [insert into jobs.accounts_acnt (active,created_at,email,name,owner_id,updated_at,acnt_id) values (?,?,?,?,?,?,?)]; constraint [accounts_acnt_owner_id_fkey]
        at com.kgignatyev.fss_svc.api.fss_client.v1.apis.AccountsServiceV1Api.createAccount(AccountsServiceV1Api.kt:77)
        at com.kgignatyev.fss.service.acceptance.account.AccountSteps.current_user_can_create_a_new_account_with_the_name(AccountSteps.kt:54)
        at *.current user can create a new account with the name "Test1"(classpath:scenarios/account-management.feature:8)
```

- **And** current user can see data of account "Test1"
- **And** current user is stored in the system

This is not good!
We are leaking backend details - will not pass a security audit, or worse- hackers will take advantage of it.

Easy fix with SpringBoot exception handler:
- log everything on server once;
- supply minimum details to client;

```
17  class ExceptionControllerAdvices {
26
27      @ExceptionHandler(value = [IllegalStateException::class, IllegalArgumentException::class,
28          Exception::class, RuntimeException::class, NoHandlerFoundException::class,
29          DataIntegrityViolationException::class, SQLException::class])
30      fun ourExceptionsHandler(exception: Exception): ResponseEntity<Any> {
31          logger.error("Exception: ${exception.message}", exception)
32          val status = when (exception) {
33              is IllegalStateException -> HttpStatus.PRECONDITION_FAILED
34              is IllegalArgumentException -> HttpStatus.BAD_REQUEST
35              else -> HttpStatus.INTERNAL_SERVER_ERROR
36          }
37          val message = when( exception ){
38              is SQLException -> "DB Error"
39              is DataIntegrityViolationException -> "Data Integrity Violation"
40              else -> "${exception.javaClass.name}: ${exception.message?: " without message"}"
41          }
42          return ResponseEntity(message, status)
43      }
```

Then current user can create a new account with the name "Test1"

```
com.kgignatyev.fss_svc.api.fss_client.v1.infrastructure.ServerException: Server error : 500  Data Integrity Violation
        at com.kgignatyev.fss_svc.api.fss_client.v1.apis.AccountsServiceV1Api.createAccount(AccountsServiceV1Api.kt:77)
        at com.kgignatyev.fss.service.acceptance.account.AccountSteps.current_user_can_create_a_new_account_with_the_name(AccountSteps.kt:54)
        at *.current user can create a new account with the name "Test1"(classpath:scenarios/account-management.feature:8)
```

And current user can see data of account "Test1"

```
@Autowired
lateinit var tracer: Tracer
```

Much better but we do not have a correlation ID which would benefit troubleshooting, lets add it.

```
49  val correlationID = tracer.currentSpan()?.context()?.traceId() ?: "no-trace-id"
50  return ResponseEntity( body: "$message - $correlationID", status)
```

Then current user can create a new account with the name "Test1"

```
com.kgignatyev.fss_svc.api.fss_client.v1.infrastructure.ServerException: Server error : 500  Data Integrity Violation - 6658ee7e3df2eed370723cec320b55f6
        at com.kgignatyev.fss_svc.api.fss_client.v1.apis.AccountsServiceV1Api.createAccount(AccountsServiceV1Api.kt:77)
        at com.kgignatyev.fss.service.acceptance.account.AccountSteps.current_user_can_create_a_new_account_with_the_name(AccountSteps.kt:54)
        at *.current user can create a new account with the name "Test1"(classpath:scenarios/account-management.feature:8)
```

ATDD - can be used as ‘unit tests’ to debug a feature

Running all scenarios can take a long time which is inconvenient when we focus on one scenario.

Luckily we can use tags to focus:

```
./run-tests.sh --tags @create-account
```

```
2  Feature: Account management
3
4  @create-account
5  Scenario Outline: Create a new account
6    Given user "<userName>" is logged in
7    And current user does not have an account
8    And current user is not stored in the system
9    Then current user can create a new account with the name "<accountName>"
10   And current user can see data of account "<accountName>"
11   And current user is stored in the system
12
13   Examples:
14     | userName  | accountName |
15     | testUser1 | Test1       |
```

2 FAILED

15 seconds ago	2.16 seconds
last run	duration

And because it is just code we can put breakpoints and debug both sides: test, and code under test!

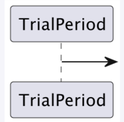
AI proofing and AI utilization

LeadsAcquisitionWorkflow.kt | PaymentsWorkflow.kt | PaymentsWorkflow.plantuml

Exceptions.kt

1:1 All Pages cached

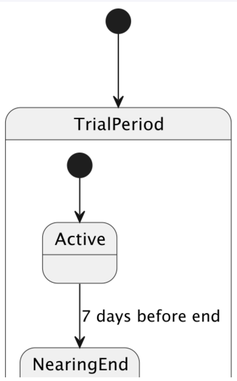
simple state diagram that starts with trial period then reminds about payments if payment details are not available, expired, or there is failure using them



Follow-up message | Accept All | Discard All | 1/1

1:1 All Pages 3662ms

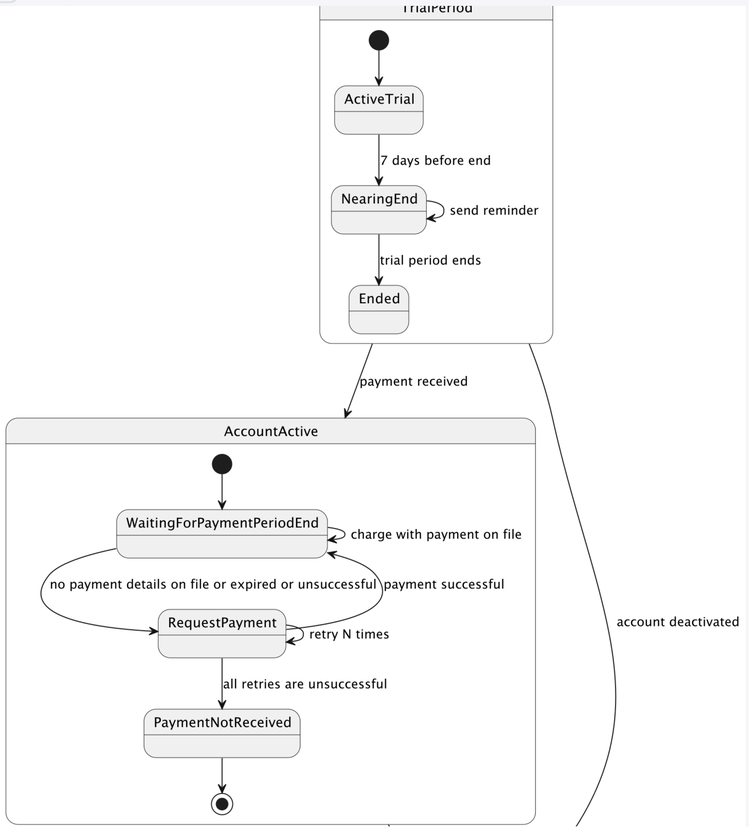
```
@startuml
[*] --> TrialPeriod

state TrialPeriod {
    [*] --> Active
    Active --> NearingEnd : 7 days before end
    NearingEnd --> Ended : trial period ends
}

TrialPeriod --> RequestPaymentDetails : trial ended

state PaymentProcessing {
    [*] --> AttemptCharge
    AttemptCharge --> Success : payment successful
    AttemptCharge --> Failed : payment failed
}

state RequestPaymentDetails {
    [*] --> WaitingForDetails
    WaitingForDetails --> Reminder : 3 days no respons
```



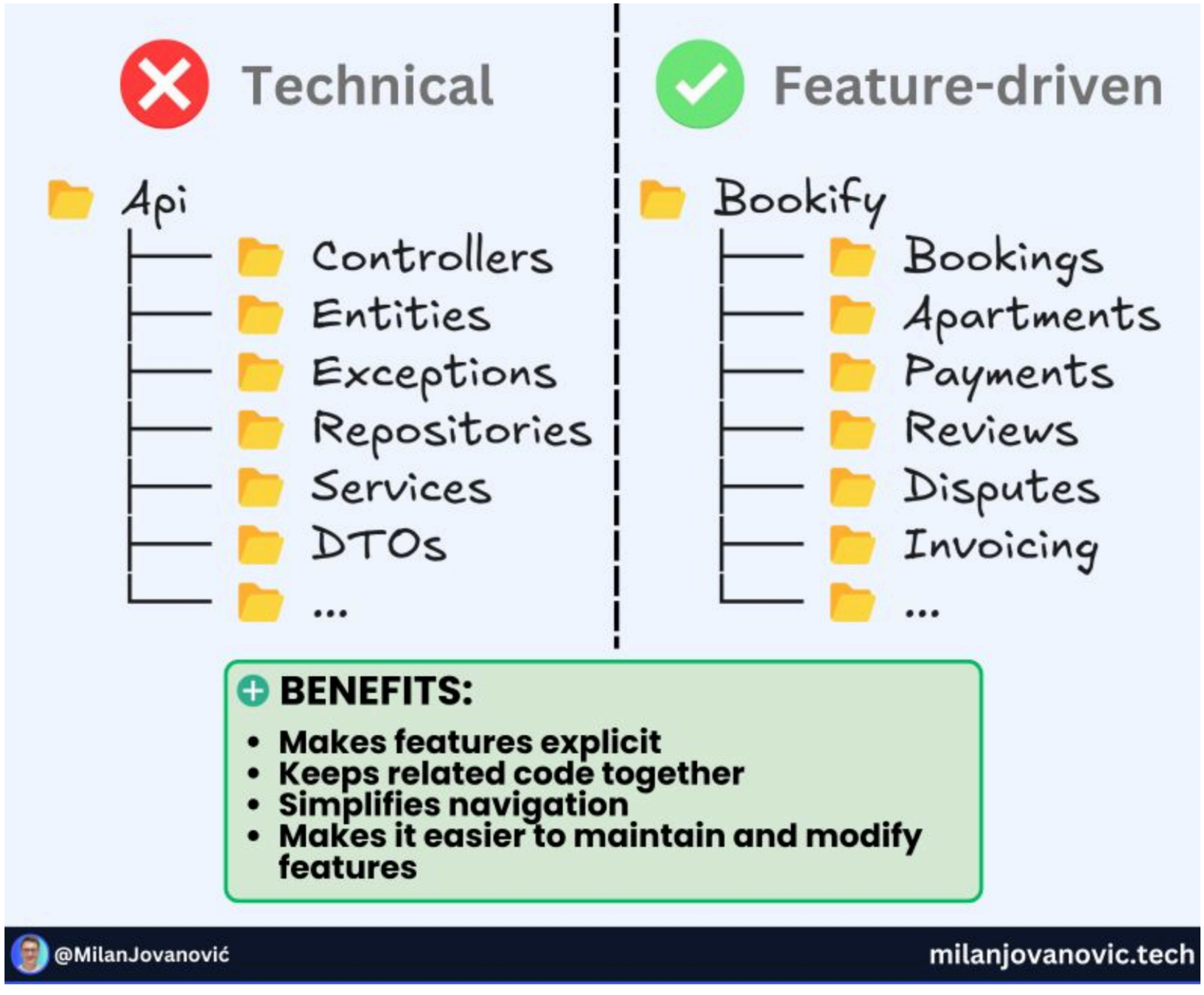
PaymentsWorkflow.plantuml

1:1 All Pages 51ms

```
ialPeriod --> AccountActive: payment received
ialPeriod --> [*]  : account deactivated

ate AccountActive {
  [*] --> WaitingForPaymentPeriodEnd

  WaitingForPaymentPeriodEnd --> WaitingForPaymentPer
  WaitingForPaymentPeriodEnd --> RequestPayment: no p
  RequestPayment --> WaitingForPaymentPeriodEnd: paym
  RequestPayment --> RequestPayment: retry N times
  RequestPayment --> PaymentNotReceived: all retries
  PaymentNotReceived --> [*]


countActive--> [*]  : account deactivated
nduml
```



Modulith! - let's do MVP as quickly as possible and do not compromise on expandability



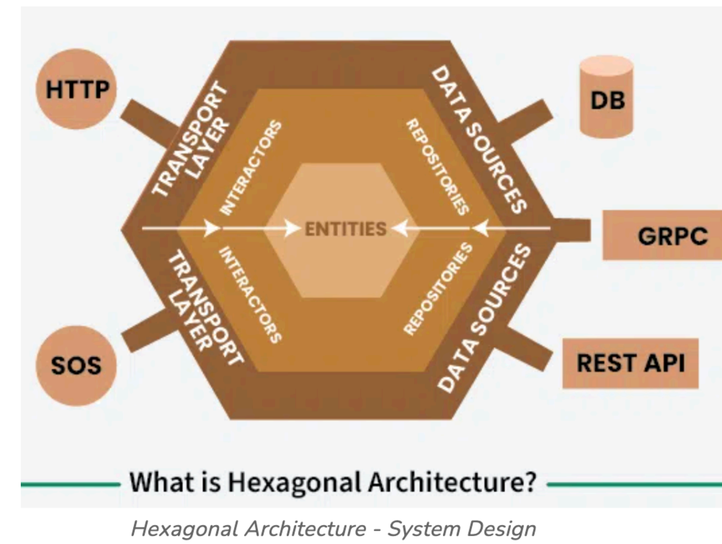
https://blog.devgenius.io/building-modular-apps-with-spring-modulith-cbb2b4aa6415

Modulith! - let's do MVP as quickly as possible and do not compromise on expandability

Onion/Hexagonal Architecture applied per module == Modulith

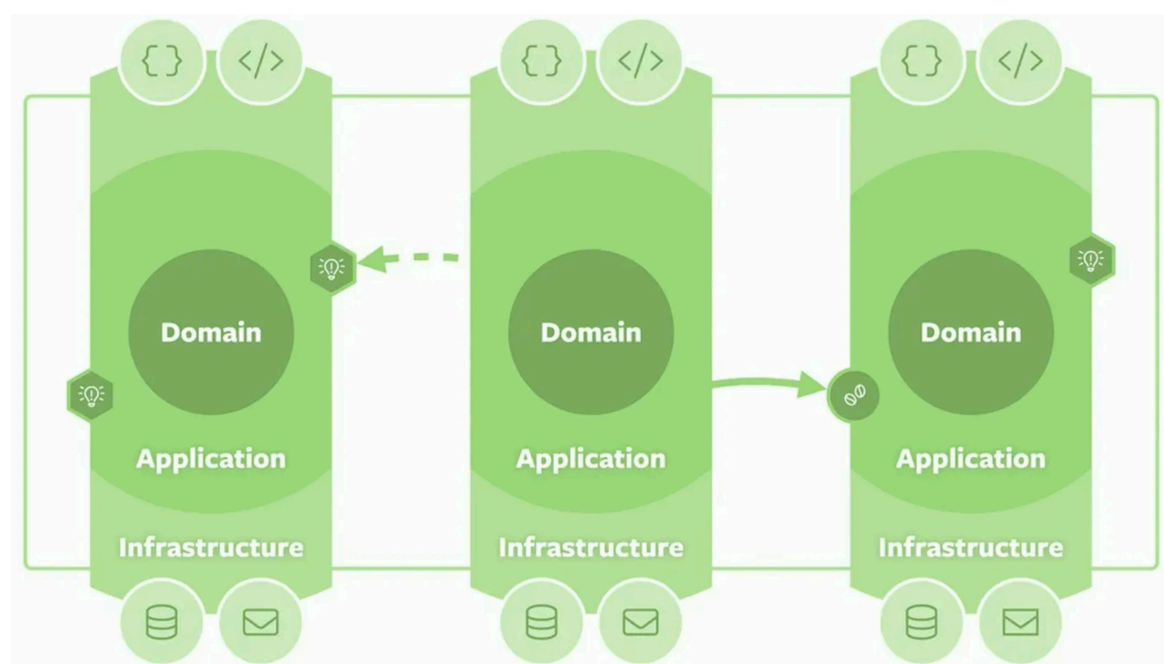
```
kotlin
  com.kgignatyev.fss.service
    accounts
    common
      api
        api_helpers.kt
        exception_handlers.kt
        package-info.java
        V1StatusHelpers
      data
      events
        CrudEvent
        CrudEventType
        package-info.java
    company
    job
      conv
      storage
      svc
      Job
      JobEvent
      JobEventEvent
      JobEventService
      JobService
      JobsServiceV1ApiImpl
    security
    Exceptions.kt
    FssService
    LoggingCongig.kt
```

https://www.geeksforgeeks.org/system-design/hexagonal-architecture-system-design/#



Spring Modulith

https://spring.io/projects/spring-modulith



https://ishansoninitj.medium.com/creating-a-multi-module-monolith-using-spring-modulith-f83053736762

Modulith - evolution from single developer to multiple teams!

Stage 1

MVP - 1 or 2 developers

Angular + DevExtreme

DevExtreme by DevExpress

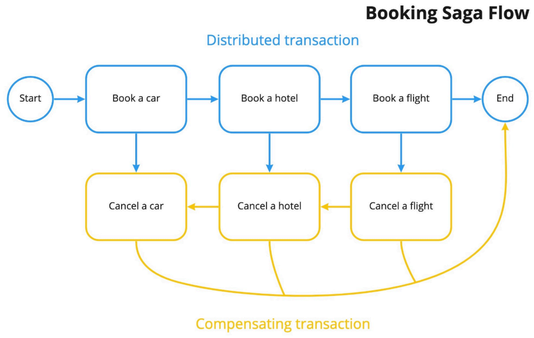
UI builders FlutterFlow and Bubble etc.

Internet

Modulith 1.1.3

HAZELCAST

Temporal
Build invincible applications



Postges

Stage ... N

any number of teams - 100's of developers

Native Mobile clients

Temporal **Cloud**

Separate deployables per responsibility

Refactoring (groups of) modules into separate services

kubernetes

Periodic Modularity tests help us to find architectural problems early

```
[INFO] Results:
[INFO]
[ERROR] Errors:
[ERROR]   ModularityTest.verifiesModularStructure:13 » Violations - Cycle detected: Slice com.kgignatyev.fss.service.accounts ->
                Slice com.kgignatyev.fss.service.security ->
                Slice com.kgignatyev.fss.service.accounts
  1. Dependencies of Slice com.kgignatyev.fss.service.accounts
    - Constructor <com.kgignatyev.fss.service.accounts.svc.AccountsSvcImpl.<init>(com.kgignatyev.fss.service.accounts.storage.AccountsRepo,
ingframework.context.ApplicationEventPublisher, com.kgignatyev.fss.service.security.SecuritySvc)> has parameter of type <com.kgignatyev.fss
.security.SecuritySvc> in (AccountsSvcImpl.kt:0)
    - Field <com.kgignatyev.fss.service.accounts.svc.AccountsSvcImpl.securitySvc> has type <com.kgignatyev.fss.service.security.SecuritySvc
countsSvcImpl.kt:0)
```

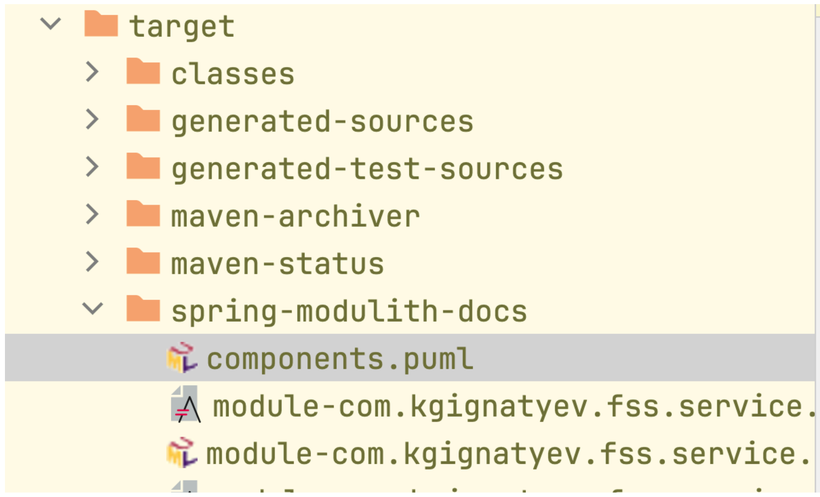
```
  2. Dependencies of Slice com.kgignatyev.fss.service.security
    - Method <com.kgignatyev.fss.service.security.authorization.BelongsToAccountFunction.accountIdOf(java.lang.Object)> checks instanceof <com.kgig
natyev.fss.service.accounts.Account> in (BelongsToAccountFunction.kt:60)
    - Method <com.kgignatyev.fss.service.security.authorization.BelongsToAccountFunction.accountIdOf(java.lang.Object)> calls method <com.kgignatye
v.fss.service.accounts.Account.getId()> in (BelongsToAccountFunction.kt:61)
```

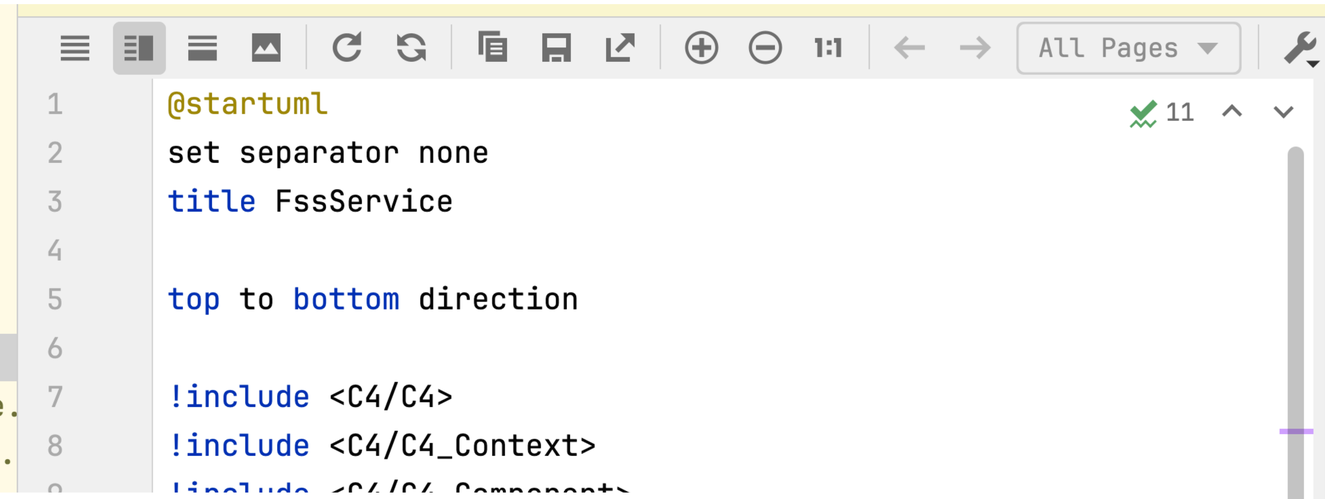
While we could allow circular dependencies arguing that Authorization system should know everything, we better introduce proper abstractions

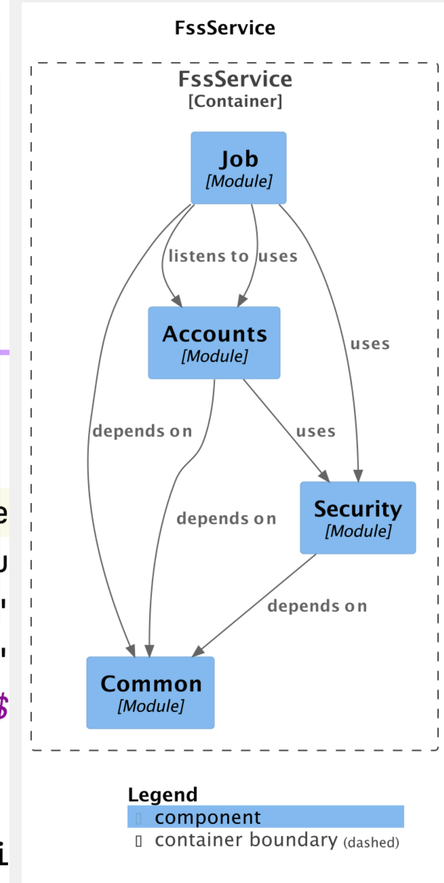
```kotlin
@Entity   kgignatyev *
@Table(name = "accounts_acnt")
class Account:Securable {
    @Id
    @GeneratedValue(strategy= GenerationType.UUID)
    @Column(name = "acnt_id")
    var id = ""
```

```kotlin
enum class SecurableType( val tName:String) {  new *
    ACCOUNT( tName: "Account"),
    JOB( tName: "Job"),
    USER( tName: "User"),
}

interface Securable {  new *
    fun type(): SecurableType  new *
    fun id(): String  new *
    fun accountId(): String  new *
    fun ownerId(): String  new *
}
```

```
mvnd  test -Dtest=ModularityTest
```



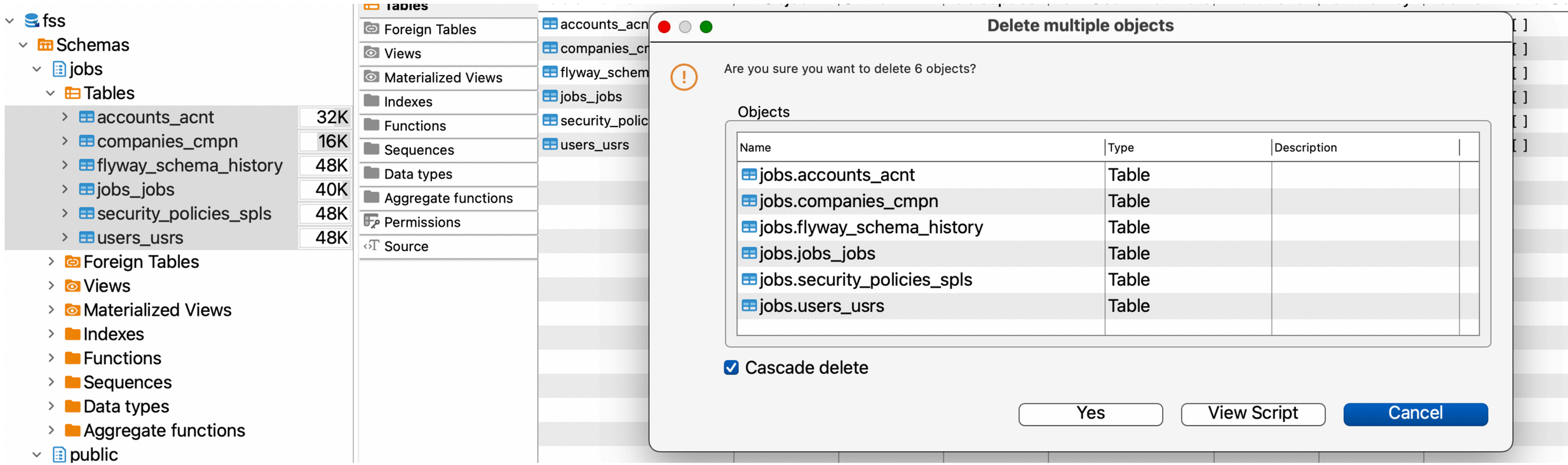




Base package	`com.kgignatyev.fss.service.accounts`
Spring components	*Controllers* • `c.k.f.s.a.AccountsSvcV1Impl` *Services* • `c.k.f.s.a.AccountsSvc` (via `c.k.f.s.a.svc.AccountsSvcImpl`) *Repositories* • `c.k.f.s.a.AccountsSvc`
Bean references	• `c.k.f.s.s.SecuritySvc` (in Security)
Aggregate roots	• `c.k.f.s.a.Account`
Published events	• `c.k.f.s.a.AccountEvent` created by: ◦ `c.k.f.s.a.svc.AccountsSvcImpl.deleteById(…)`

DB management

```
db.migration
  V1.0__initial_schema.sql
  V1.2__add_user_and_policies.sql
```



DB management for real: there is second stream of changes (data fixes) which is applied to production only

5. Maven:

Code

```
<configuration>
    <table>my_schema_history_table</table>
</configuration>
```

6. Environment Variable:

Code

```
export FLYWAY_TABLE=my_schema_history_table
```

By setting the `flyway.table` property, Flyway will use the specified name for its schema history table, instead of the default `flyway_schema_history`.

Authorization Management: ABAC! - implemented with Casbin

https://medium.com/@kgignatyev/unification-of-authorization-for-ui-and-backend-with-casbin-1523758658f7

fss_authorization_model.conf

```
[request_definition]
r = sub, obj, act

[policy_definition]
p = sub, obj, act

[policy_effect]
e = some(where (p.eft == allow))

[matchers]
m = isActive(r.sub) && hasAccessTo(p.obj, r.obj) && canDoAction( p.act, r.act)
```

	usrs_id	name	policy
1	admin kgignatyev+fss@gmail.com	admin	*, *
2	guest guest	guest	none, none
3	'dee4 1a8640ae-4417-4374-a761-c5a876d42606 kgi	self access	User/1a8640ae-4417-4374-a761-c5a876d42606, *
4	45ad 1a8640ae-4417-4374-a761-c5a876d42606 kgi	account-owner	Account/ee369832-57bd-4aea-afbe-d9d72107b002, *
5	0946 9079d3ee-7fc7-4f9b-a8c1-429fbaa930ad kgig	self access	User/9079d3ee-7fc7-4f9b-a8c1-429fbaa930ad, *
6	d8e9 9079d3ee-7fc7-4f9b-a8c1-429fbaa930ad kgig	account-owner	Account/3bd9a551-b09a-4b99-bcb5-09220045e8c1, *

Casbin

An authorization library that supports access control models like ACL, RBAC, ABAC for Golang, Java, C/C++, Node.js, Javascript, PHP, Laravel, Python, .NET (C#), Delphi, Rust, Ruby, Swift (Objective-C), Lua (OpenResty), Dart (Flutter) and Elixir

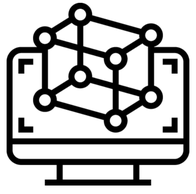
Get Started

Golang | Java | C/C++ | Node.js | Front-end js | PHP | Laravel

Python | .Net (C#) | Delphi | Rust | Ruby | Swift (Objective-C)

Lua (OpenResty, Kong, APISIX) | Dart (Flutter) | Elixir

Cloud Native (Kubernetes, Istio, Envoy, KubeSphere)



Hybrid access control models

Flexible policy storage

Cross-languages & cross-platforms