Carimbo de data/hora	Pontuação	Name and Student ID	What is the name of the file you evaluated?	How would you rate your confidence that there is a SHOTGUN SURGERY smell in the code?	Mention why (for instance, which code elements) you gave this rating.
27/04/2025 14:12:23		UFMG #44	Faker.java	1	Embora tenha outros smells, a classe não possui Shotgun Surgery. Embora ela use vários Objetos diferentes, seus métodos além de serem simples, sempre trabalham ou em retornar um objeto que foi criado na classe; ou chamar o método que faz manipulação de string, sempre retornando String. Faker não tem SS pois não usa nenhum dos retorno de forma significativa.
28/04/2025 22:17:12		UFMG #37	AssemblyProcessor	4	Many fiels and method with several branches.
28/04/2025 22:30:22		UFMG #37	Vault	5	It has many attributes and methods. Most importantly, it addresses several concerns, such as error handing, display, configurations, etc.
29/04/2025 12:32:50		UFMG #44	ClientHandler	3	É difícil saber se a classe tem problema de Shotgun Surgery avaliando a classe isoladamente. Precisaria saber quantos outros elementos no projeto dependem dessa classe.
05/05/2025 17:08:46		UFMG #6	MigrationRunController	2	Apesar de ser uma interface onde não vejo a implementação a classe está bem estruturada.
06/05/2025 21:09:25		UFMG #8	ContentType	1	Acho que somente uma classe não é suficiente para identificar este code smell.
14/05/2025 17:44:41		UFMG #5	AwtCodec	2	Cada método modifica poucas coisas, embora tenham muitos if elses dentro de cada um.
14/05/2025 17:49:25		UFMG #9	Selectable	5	Uma mudança pequena na classe Scalable teria consequências em todos os métodos.
14/05/2025 17:50:10		UFMG #12	CameraInputController	1	Single changes doesn't require many smalls modifications across the code. Despite that, other code smells were detected: Large Class (many methods, fields and even another class implemented inside the class CameraInputController) and Long Method (multiple levels of if/else statements inside the class's methods).
14/05/2025 17:50:30		UFMG #42	RandomService	3	Imagino que o método hex sem parâmetros possa causar problemas caso seja modificado. Ele retorna hex(8), ou seja, um hexadecimal aleatório de tamanho 8. Pode ser que outras classes dependam de um valor hexadecimal com um tamanho exato igual a 8.
14/05/2025 17:50:59		UFMG #16	ClientHead	3	Dispersed Coupling - Os métodos da classe ClientHead dependem em grande parte dos métodos das classes Channel e Transport
14/05/2025 17:51:15		UFMG #5	AssemblyProcessor	2	Sem saber outras classes não é possível dizer se são métodos que reimplementam outras partes de outras classes.
14/05/2025 17:54:26		UFMG #26	NodeTraversor.java	5	O método filter possui muitas verificações condicionais com if, aumentando a complexidade de entendimento do código. Seria possível, talvez, separar o código em métodos menores. Bad smell: Long Method. Além disso, faz mais chamadas de métodos de outras classes do que a si mesma, dependendo de múltiplas classes para seu funcionamento. Dispersed Coupling.
14/05/2025 17:54:52		UFMG #10	UmsMemberService	2	O código apresentado não possui nem sequer as implementações dos métodos, sendo assim, não é possível identificar um bad smell do tipo Shotgun Surgery. O arquivo entretanto, é uma classe grande, com muitos métodos e responsabilidades, contendo um badsmell de large class.
14/05/2025 17:57:34		UFMG #24	NiceSecurePasswordField	2	O código indicado não apresenta claramente um problema de Shotgun Surgery, porém os métodos da linha 73 até 107 são redundantes, e apenas repetem funcionalidades da classe SecurePasswordField, indicando prováveis problemas na configuração da herança, o que pode causar problemas desse tipo.
14/05/2025 17:57:51		UFMG #43	AnalysisContext	4	Há metodos muito semelhantes para classes que diferentes (readSomethingHolder) e além disso há muitos @Deprecated (obsoleto) indicando possível bequest
14/05/2025 17:58:03		UFMG #21	JUnitCommandLineParseResult	2	A classe JUnitCommandLineParseResult apresenta shotgun surgery porque qualquer adição ou modificação em uma opção de linha de comando exige alterações em várias outras partes do código
14/05/2025 17:58:36		UFMG #9	NiceSecurePasswordField	5	Uma mudança na classe SecurePasswordField implicaria em mudanças em praticamente todos os métodos. Acho que também tem Long Method com o construtor e o Feature Envy, porque a classe usa muitas outras classes.
14/05/2025 17:58:38		UFMG #18	ClientHead	2	No caso de ClientHead, todas as responsabilidades (gerenciamento de transporte, agendamento de timeout, clientes de namespace etc.) estão encapsuladas em uma única classe; mudanças tendem a impactar métodos internos ou adicionar novos, sem obrigar modificações em outras classes espalhadas pelo sistema.
14/05/2025 17:58:59		UFMG #26	NodeTraversor	2	INVALIDAR PRIMEIRO FORMS DE NodeTraversor.java Apesar da utilização de multiplas classes, acredito que a mudança nessa não afetará outras partes do código, portanto, acredito que a chance de ter esse bad smell em específico não é muito alta.
14/05/2025 17:59:52		UFMG #18	JSONTokener	1	Toda a lógica de parsing está concentrada nessa única classe, de forma que ajustes futuros provavelmente afetarão apenas este componente e não demandarão alterações simultâneas em múltiplas classes
14/05/2025 18:00:46		UFMG #42	QueueTest	2	Como é uma classe para teste, imagino que não vá interferir em outras classes (focando no caso do code smell shotgun surgery). Apesar disso, existem métodos longos, que a legibilidade é prejudicada, usando for loops aninhados e blocos try catch. O código nesse caso poderia ter sido fracionado em partes menores.
14/05/2025 18:01:55		UFMG #7	EntryConfig	4	O código tem um shotgun surgery por causa dos múltiplos construtores, que dificultam a manutenção
14/05/2025 18:02:47		UFMG #16	JUnitCommandLineParseResult	5	Shotgun Surgery - Não encontrado Long Method - O método String[] parseOptions(String... args) possui muitas condicionais aninhadas
14/05/2025 18:03:32		UFMG #27	CameraInputController	1	Não tem code smell porque o código concentra as alterações relacionadas à entrada de câmera em um único lugar (a própria classe CameraInputController). Se precisar mudar o comportamento da câmera, é preciso apenas mexer na classe Camera.
14/05/2025 18:05:22		UFMG #14	UmsMemberService	4	Essa função pode espalhar muito os erros existentes já que ela envia muitas informações
14/05/2025 18:05:38		UFMG #30	XxlJobInfoDao	4	Muitos métodos com parâmetros similares; mudanças em critérios exigem alterar vários pontos.
14/05/2025 18:05:42		UFMG #10	Excluder	5	O código apresenta o badsmell Shotgun Surgery, isso fica claro pelos varios codigos similares, onde uma variavel "Excluder result" recebe o metodo "clone()" faz alguma modificacao e retorna a variavel "result". Também é possivel identificar os badsmells tipo Large Class e Long Method.
14/05/2025 18:06:38		UFMG #36	TextEditorUtils	4	Analisando o código, pensei que o método enableHostEditorKeyBindings utiliza um método privado de AbstractTextEditor e, se a API interna desse Eclipse mudar, a funcionalidade irá quebrar e outras que dependem da mesma tambem. Caso essas funcionalidades sejam amplamente utilizadas no DBeaver e as correções em TextEditorUtils alterem suas interfaces, múltiplas classes consumidoras poderiam necessitar de pequenos ajustes.
14/05/2025 18:07:06		UFMG #1	JSONTokener	5	A presença de diversos métodos next() e nextTo() requer que, caso uma mudança no funcionamento desta regra seja exigida, vários métodos vão necessitar de modificações pequenas, o que caracteriza um smell de Shotgun Surgery. Além disso, o método nextString() possui um switch case relativamente grande que talvez possa ser reduzido, caracterizando um smell de long method
14/05/2025 18:07:26		UFMG #4	ParserConfig.java	5	There is absolutely a code smell, because by the size of this class , tracking errors is impossible in this class.
14/05/2025 18:09:01		UFMG #16	Lexeme	5	Shotgun Surgery - Não Encontrado Long Method - O método "public int compareTo(Lexeme other)" possui muitas condicionais aninhadas
14/05/2025 18:09:49		UFMG #12	Lifecycle	1	It's seems that the responsibilities are well organized within Lifecycle class. Thus single changes doesn't seem to require many small modifications across the code.
14/05/2025 18:10:02		UFMG #43	Lifecycle	2	Apresenta várias mini cópias de código, como os diversos catch (possivelmente justificáveis dependendo de como são tratadas no restante do código)
14/05/2025 18:10:14		UFMG #18	XxlJobInfoDao	1	Pois este antipadrão ocorre quando uma única mudança de requisito exige modificações pequenas em múltiplas classes diferentes ao mesmo tempo. No caso de um DAO, as alterações em consultas ou parâmetros envolvem essencialmente mudanças na própria interface e nos mapeamentos associados, mas não se espalham por diversas classes arbitrárias do sistema. Portanto, não há indícios de que um ajuste em lógica de paginação, critérios de busca ou modelo de dados exigiria "cirurgias" simultâneas em muitas classes distintas, característica central do Shotgun Surgery.
14/05/2025 18:10:39		UFMG #41	JUnitCommandLineParseResult	2	Apresenta acumulo de responsabilidade
14/05/2025 18:12:31		UFMG #14	RandomService	5	O código da classe `RandomService` tem o cheiro de **Shotgun Surgery**, pois para adicionar ou modificar funcionalidades é necessário alterar várias partes do código espalhadas. Para melhorar, seria ideal separar as funcionalidades de geração de números aleatórios em classes distintas, como uma para inteiros, outra para longos, etc. Isso centraliza responsabilidades, facilita a manutenção e evita alterações em várias partes do código ao mesmo tempo.
14/05/2025 18:12:35		UFMG #39	FlowControllerV1	3	As rotas que adicionam novas regras afetam a validação de outras rotas, então uma alteração se espalharia para várias. Não tenho certeza se isso seria um code smell ou somente uma funcionalidade planejada.
14/05/2025 18:13:11		UFMG #28	RequestLimiter	5	A lógica " this.qpsAllowed = qpsAllowed;" é implementada 2 vezes, em contextos diferentes, o que indica uma falta de organização das responsabilidades, e sobretudo, um shotgun surgery.
14/05/2025 18:13:19		UFMG #29	Selectable	5	Apresenta o shotgun, porque sempre que uma mudança no comportamento de seleção é necessária várias classes precisam ser modificadas ao mesmo tempo para refletir essa mudança.
14/05/2025 18:14:31		UFMG #34	UmsMemberService	4	Minha confiança é 4. Apesar de ser uma interface, UmsMemberService pode levar a Shotgun Surgery se considerarmos o impacto de mudanças em suas definições. Por exemplo, se um novo parâmetro obrigatório fosse adicionado ao método register (como data de nascimento), essa alteração exigiria modificações não apenas nesta interface, mas em todas as suas classes de implementação concretas e em todos os locais do código que chamam esse método, espalhando a necessidade de ajustes por múltiplos arquivo
14/05/2025 18:14:33		UFMG #2	TextEditorUtils	3	O método enableHostEditorKeyBindings usa reflexão para acessar um método privado da classe AbstractTextEditor. Isso torna a manutenção do código mais difícil, pois mudanças na implementação da classe AbstractTextEditor exigirão alterações em vários pontos que usam esse mecanismo. Assim, alterações na forma como ações do editor são ativadas ou desativadas causariam impactos espalhados, dificultando a manutenção e a evolução do código.
14/05/2025 18:14:54		UFMG #13	AnalysisContext.java	1	Não tem nada de code smell na minha opnião
14/05/2025 18:16:40		UFMG #25	eval-llms-code-smells /ClientHead	2	Tive dificuldade de entender o código, por isso também tive dificuldade de avaliar se tem o code smell shotgun surgery. Por a classe possuir muitas interfaces com várias outras classes e executar muitas funcionalidades, acredito que possa ter esse code smell, porém não sei afirmar com certeza
14/05/2025 18:18:52		UFMG #38	UmsMemberService	1	O que vejo: 1) É só uma interface, sem código executável. Não existe método longo, switch gigante, nada disso. 2) Ela agrupa funções ligadas a "membro/usuário": buscar, registrar, login, token – tudo parece caber no mesmo contexto, então não senti falta de separar em outras interfaces. 3) Lista de parâmetros é curta; o pior caso tem 4 argumentos (register), bem abaixo do "6+" que na aula foi o valor usado para Long Parameter List. 4)Se quiser pegar no pé, dá pra dizer que mistura autenticação (token) com cadastro numa única interface (levíssima suspeita de Divergent Change), mas honestamente é detalhe. Resumindo: quase nenhum bad smell perceptível; dei 1 pela leve mistura de responsabilidades, mas não chega a preocupar.
14/05/2025 18:19:13		UFMG #22	UmsMemberService.java	4	Código muito estranho, qualquer alteração feita em um dos métodos de UmsMemberService pode causar erros em códigos que chamam a este
14/05/2025 18:23:02		UFMG #26	AssemblyProcessor	5	Por utilizar diversas verificações condicionais sequenciais nos métodos, a alteração em alguma delas provavelmente levaria a alteração de condição de várias outras, num efeito cascata. A longa lista de atributos também pode contribuir para eses bad smell.
14/05/2025 18:25:38		UFMG #31	CameraInputController	5	O código possui vários code smells, como uma classe longa com muitos atributos que dependem de classes externas, e isso pode levar ao Shotgun Surgery
14/05/2025 18:26:43		UFMG #13	XxlJobInfoDao	1	Há apenas métodos simples que são facilmente identificáveis ao serem utilizados, portanto, alterações nessa classe provavelmente seriam facilmente tratadas.
14/05/2025 18:29:01		UFMG #38	RandomService	2	This class does not have the Shotgun Surgery code smell. It encapsulates randomness in one place, and changes to random generation logic would be localized here, not across multiple classes.
14/05/2025 18:29:17		UFMG #28	Lexeme	5	A mesma lógica "this.lexemeType = lexemeType;" é usada em diferentes contextos, causando um problema de responsabilidades no código.
14/05/2025 18:29:38		UFMG #24	EmbedServer	5	O código possui diversas mensagens de log diferentes definidas no próprio código. Nessas mensagens há diversas substrings repetidas, como "invalid request", "xxl-job remoting server ". Uma alteração nessas substrings teria como consequência uma necessidade de alteração em todas as strings de log que possuem essa substring.
14/05/2025 18:32:20		UFMG #31	QueueTest	2	O código é simples, a alteração de um teste não parece exigir mudanças em outras partes do código
14/05/2025 18:32:41		UFMG #16	AnalysisContext	1	Shotgun Surgery - Não encontrado
14/05/2025 18:33:40		UFMG #36	Lexeme	4	Acredito que o problema de Shotgun Surgery está problema, porque alterações nas definições iniciais, especialmente nos lexemeType, apresentam um risco significativo para o resto do código e outros métodos. A adição, remoção ou modificação de um tipo de lexema exigiria atualizações não só nesta classe, mas também em múltiplas outras classes do sistema que dependem da lógica de tratamento desses tipos.
14/05/2025 18:33:56		UFMG #19	NiceSecurePasswordField	5	Caso fosse mudado algo na classe PasswordField, seriam necessárias mudanças imediatas em NiceSecurePasswordField também.
14/05/2025 18:34:10		UFMG #22	DataFormatter.java	4	Não há o codigo smell fornecido, mas há long method
14/05/2025 18:37:37		UFMG #16	Excluder	1	Não encontrado nenhum bad smell
14/05/2025 18:40:32		UFMG #35	UmsMemberService	2	Não creio que se enquadra nesse code smell pois considero a classe e métodos independentes.
14/05/2025 18:41:58		UFMG #41	TextEditorUtils	1	Não encontrei evidências de smells do tipo Shotgun Surgery.
14/05/2025 18:42:07		UFMG #45	Lifecycle	2	O código apresenta um long method. Não apresenta shotgun surgery.

Carimbo de data/hora	Pontuação	Name and Student ID	What is the name of the file you evaluated?	How would you rate your confidence that there is a SHOTGUN SURGERY smell in the code?	Mention why (for instance, which code elements) you gave this rating.
14/05/2025 18:44:21		UFMG #1	JavassistProxyFactory	4	Este código é terrível e extremamente difícil de debuggar e dar manutenção por estar totalmente escrito com reflection, porém caso haja necessidade de alterar algo nele, creio que você só irá fazer mudanças neste arquivo e em pontos específicos (porém, difícil de localizar). Agora, caso você mexa no classe que está sendo chamada por reflection, então com certeza você terá bastante retrabalho, especialmente porque o código não irá quebrar em compile-time, você terá que testar o código inteiro praticamente para achar os problemas em run-time e então mudar o código.
27/05/2025 11:39:48		UFMG #27	ParserConfig	5	Alterações como adicionar uma nova regra de desserialização ou configurar uma propriedade exigem modificações espalhadas por diversos métodos e estruturas da classe, como initDeserializers, configFromPropety, checkAutoType e blocos static. Além disso, há um alto acoplamento com múltiplas classes auxiliares.
04/06/2025 19:46:15		UFOP #8	SaTokenConfig	5	A classe utiliza dos nomes dados as variáveis em diversos lugares, como nos métodos de get, set e toString. Se houvesse uma mudança de negócio, por exemplo substituir o verbo is por has, seria necessário alterar por toda a classe.
04/06/2025 19:47:25		UFOP #19	TokenQueue	2	Não consegui notar nenhum code smell muito claro relacionado a Shotgun Surgery. Entretanto, acredito que possa ser uma large class.
04/06/2025 19:48:27		UFOP #31	RedissonObject.java	3	código poderia estar mais resumido, exemplo prefixName e sufixName desnecessário, é pedido para deletar o arquivo e depois restaurar.
04/06/2025 19:50:50		UFOP #23	JSONTokener	4	Várias classes e seus métodos
04/06/2025 19:54:21		UFOP #8	HikariPool	2	O fato da classe ser grande pode gerar o smell de shotgun surgery, porém não identifiquei nenhum outro motivo claro.
04/06/2025 20:00:26		UFOP #23	HikariPool.	1	Por ter vários private por ser característica do large class
04/06/2025 20:02:39		UFOP #10	ParamFlowChecker	5	Aparentemente existe um smell tipo shotgun surgery nesse trecho de código, pois existem alguns métodos que são basicamente uma cópia de um método anterior e uma alteração na lógica de um método resultaria na alteração de vários outros
04/06/2025 20:02:44		UFOP #23	NacosDiscoveryProperties	5	Identifiquei mudanças múltiplas no código
04/06/2025 20:06:50		UFOP #11	HikariPool	3	A classe principal não parece apresentar shotgun surgery, porém algumas subclasses como HikariPool e HikariPool apresentam os mesmos métodos e em caso de uma manutenção geraria retrabalho e redundancia.
04/06/2025 20:06:53		UFOP #4	HikariConfig	2	Atribui 2, pois não consegui identificar code smell do tipo Shotgun Surgery (posso estar errada). O que me chamou atenção foi o tamanho do código e o grande número de variáveis, creio que por conta disso a identificação de smells do tipo Shotgun Surgery ficou dificultada.
04/06/2025 20:08:13		UFOP #22	PooledDataSource.java	4	Acredito que sim, pois uma simples mudança no método expectedConnectionTypeCode vai requirir diversas modificações em outros lugares do código
04/06/2025 20:08:16		UFOP #8	BrokerController	4	Além de ser uma classe muito grande e com diversas injeções de dependências, ela tem diversas partes, como instanciar um BrokerFixedThreadPoolExecutor, um scheduleAtFixedRate, que utilizam das mesma estrutura. Caso seja necessário alguma alteração de negócio por exemplo, seria necessário alterar por toda a classe.
04/06/2025 20:09:31		UFOP #7	HttpServerTest	3	Apresenta em quantidade moderada shotgun surgery, uma vez que os métodos de teste testMultipartFormData e testTempFileInterface possuem lógica quase idêntica para configurar o servidor HTTP, enviar requisições multipart e verificar respostas.
04/06/2025 20:10:36		UFOP #3	ParamFlowChecker	3	Embora organize bem as checagens principais, alguns métodos como passDefaultLocalCheck são longos e complexos, com muitas decisões e loops, o que pode dificultar o entendimento e testes.
04/06/2025 20:12:20		UFOP #7	BitMatrix	1	Não apresenta nada de shotgun surgery, uma vez que os metodos estão bem especificados e são quase independentes.
04/06/2025 20:12:46		UFOP #12	BitMatrix	1	Não fui capaz de encontrar um code smell similar ao do exemplo. Portanto, baseando nos conhecimentos que possuo, acredito que não tenha nenhum code smell
04/06/2025 20:13:11		UFOP #17	TokenQueue.java	1	Não encontrei evidências que justifiquem essa smell
04/06/2025 20:16:14		UFOP #12	HttpServerTest	4	Acredito que possa exisitr, pois há uma duplicação nos testes testMultipartFormData e testTempFileInterface
04/06/2025 20:16:41		UFOP #9	RedissonObject	4	Este código apresenta características do code smell Shotgun Surgery, então eu diria que sim, possui este code smell. Pois uma mudança requer muitas modificações em várias outras classes, em outras partes do código. Isso ocasiona uma manutenção cansativa, tendo que localizar todas implementações afetadas.
04/06/2025 20:19:44		UFOP #25	RedissonObject	5	ao longo do código é possível ver que ele possui muitas substituições (override)
04/06/2025 20:20:25		UFOP #2	HikariConfig	3	A classe possui certas lógicas que caso alteradas podem trazer a necessidade de mudança em pontos do código, porém não o suficiente para ser notável.
04/06/2025 20:20:28		UFOP #21	Closer	4	Acredito que o código tem sim uma classificação de Shotgun Surgery, pois existem algumas funções que são implementadas que fazem a mesma coisa, só que de forma parecida para cada função.
04/06/2025 20:21:08		UFOP #27	Closer	4	A lógica de fechamento está espalhada em múltiplos lugares interdependentes.
04/06/2025 20:21:09		UFOP #17	shotgun_surgery.java	3	Pequenas mudanças nessa classe não provocarão alterações de comp ortamento críticas em outras classes
04/06/2025 20:21:37		UFOP #18	HttpServerTest.java	3	Testes com muitas assercoes, metodos testa varias coisas em uma unica classe.
04/06/2025 20:22:15		UFOP #5	InterceptorProcessor.java	1	Não vi nada suspeito, apenas um código que importou muita coisa, mas não parece que outros dependem dele não
04/06/2025 20:22:46		UFOP #28	NacosDiscoveryProperties	1	O método não se encaixa pois os setters e getters estão bem definidos.
04/06/2025 20:23:17		UFOP #6	TokenQueue	5	Existem vários métodos que são exatamente idênticos, contendo pequenas alterações para se adequar ao parâmetro, o que pode acarretar em vários erros ao realizar alguma mudança.
04/06/2025 20:23:17		UFOP #15	Connection	3	Difícil identificação, mas considero que uma simples alteração (por exemplo, adicionar um novo campo) força edições em vários locais.

04/06/2025 20:23:34		UFOP #24	Closer	4	Pois possui os seguintes métodos:
					@Override
					public void suppress(Closeable closeable, Throwable thrown, Throwable suppressed) {
					// log to the same place as Closeables
					Closeables.logger.log(
					Level.WARNING, "Suppressing exception thrown when closing " + closeable, suppressed);
					}
					@Override
					public void suppress(Closeable closeable, Throwable thrown, Throwable suppressed) {
					// ensure no exceptions from addSuppressed
					if (thrown == suppressed) {
					return;
					}
					try {
					addSuppressed.invoke(thrown, suppressed);
					} catch (Throwable e) {
					// if, somehow, IllegalAccessException or another exception is thrown, fall back to logging
					LoggingSuppressor.INSTANCE.suppress(closeable, thrown, suppressed);
					}
					}

04/06/2025 20:25:44		UFOP #6	ParamFlowChecker	2	Embora existam métodos com semelhanças, muitos acabam tendo bastante diferenças em seu escopo, o que evita um efeito em cascada, ao realizar mudanças.
04/06/2025 20:26:02		UFOP #5	HikariConfig.java	5	Mudanças na lógica de configuração, nomes de propriedades ou comportamentos de propriedades no código apresentado frequentemente exigem mudanças em muitos outros arquivos. Ele parece um hub de configuração
04/06/2025 20:26:39		UFOP #21	Connection	5	Acredito que está sim classificado como code smell do tipo Shotgun Surgery, pois existem vários métodos iguais, mas que estão sendo feitos de forma específica dependendo do contexto, dificultando a manutenção.
04/06/2025 20:29:04		UFOP #17	PooledDataSource.java	4	Mudanças específicas nesse código podem causar reações não esperadas em outras classes do sistema
04/06/2025 20:32:08		UFOP #33	NacosDiscoveryProperties.java	3	No método overrideFromEnv, caso ocorra alguma mudança na sua forma de funcionamento pode ser que a aplicação quebre devido ao demasiado uso de condicionais.
04/06/2025 20:34:09		UFOP #33	SaTokenConfig.java	1	Não foi possível identificar algum code smell do tipo Shotgun Surgery. Posso estar errado, mas em primeira analise não consegui visualizar.
04/06/2025 20:34:20		UFOP #26	SaTokenConfig	1	Não é um caso de Shotgun Surgery pois os métodos estão centralizados facilitando a coesão e manutenção.
04/06/2025 20:35:06		UFOP #30	InterceptorProcessor	2	A classe InterceptorProcessor necessita de poucas edições para resolver mudanças simples.
04/06/2025 20:35:16		UFOP #16	InterceptorProcessor	1	Acredito que não possua o code smell SHOTGUN SURGERY, o código não apresentar estar com responsabilidades mal organizadas.
04/06/2025 20:39:32		UFOP #26	BrokerController	3	É um caso de Shotgun Surgery pois a classe depende de diversos métodos, e classes externas.
04/06/2025 20:42:00		UFOP #16	HikariPool	3	O código possui SHOTGUN SURGERY, pois em alguns momentos ele passa alguns valores inteiros como parametros para alguns métodos, como uietlySleep(MILLISECONDS.toMillis(100)
04/06/2025 20:47:43		UFOP #29	TokenQueue	2	Acredito que não haja shotgun surgery, pois as responsabilidades dos metodos esta bem organizada.