Codes extracted from C4 Interview Question (Use of code smell detection tools)									
Interviewee	**Raw answer (full)**	**Text snippet 1**		**Text snippet 2**		**Text snippet 3**		**Text snippet 4**	
		Raw answer	**Code**	**Raw answer**	**Code**	**Raw answer**	**Code**	**Raw answer**	**Code**
I1	A1: Yes. The tools may vary depending on the project, but it's very common to commits and MRs be subject to a CircleCI pipeline or similar tool in which we include a step that runs a Linter in the codebase A2: CircleCI, Github Actions and custom linters for the language, in the case of Golang for example: golangci-lint A3: I only run linters and refactor any code smell that might be pointed out after the code is done (sometimes, only after the code was tested by the unit tests); I do not run any linting or any testing whatsoever for code I consume (third party, open source projects, etc.) as part of a personal policy of not maintaining code that do not belong to me and/or I did not produce. If I do not trust the code I'm using or consuming, then I shouldn't be using it	Yes [...] it's very common to commits [...] be subject to a CircleCI pipeline or similar tool	Automated detection on commits	Yes [...] it's very common to [...] MRs be subject to a CircleCI pipeline or similar tool	Automated detection on MRs (merge requests)	I only run linters and refactor any code smell that might be pointed out after the code is done (sometimes, only after the code was tested by the unit tests)	Automated detection on code before testing	I only run linters and refactor any code smell that might be pointed out after the code is done (sometimes, only after the code was tested by the unit tests)	Automated detection on tested code
I2	A1: I use ESLint with most default rules. A2: I run it while producing code. I never thought about doing it in code I consume. A3: I consider code as done only after any smells communicated by ESLint are resolved, so I don't run it again after that. However, I do run it again while reviewing code done by teammates to make sure this practice is being followed.	I run it while producing code.	Automated detection on produced code	[...] I do run it again while reviewing code done by teammates to make sure this practice is being followed	Automated detection during code review				
I3	A1: (From C1) Yes, but we are currently not using any tools for analyzing it A2: Not encouraged by the teams I currently work with. This would require some collective effort for addressing the detected issues A3: Maybe, but I believe leaderships tend to prioritize other good practices first, like testing. For personal projects, tools like SonarQube usually require some setup that are an overkill for the project size	Not encouraged by the teams I currently work with. This would require some collective effort for addressing the detected issues	Automated detection not encouraged by the team	Not encouraged by the teams I currently work with. This would require some collective effort for addressing the detected issues	Automated detection requires collective effort	Maybe, but I believe leaderships tend to prioritize other good practices first, like testing	Code smells detection is not a priority	[...] For personal projects, tools like SonarQube usually require some setup that are an overkill for the project size	Tool setup is not worth it

I4	A1: We use tools such as GitLab to review each other's code, so that the code we produce is readable and understood to everyone on the team. A2: We use both GitLab and multiple StyleLint plugins to detect bad practices while coding with TypeScript.	We use tools such as GitLab to review each other's code, so that the code we produce is readable and understood to everyone on the team	Automated detection during code review	We use both GitLab and multiple StyleLint plugins to detect bad practices while coding with TypeScript.	Automated detection on produced code				
I5	A1: No. I only use ESlint and Prettier. Other projects use SonarQube, but not mine. A2: Not anything in particular. Because as I mentioned, Im use to work on projects that already have some time running, so I don't know exactly why. Maybe company culture	[...] Maybe company culture	Company culture						
I6	A1: Yes A2: Sonarqube A3: I have sonar lint into my IDE which shows to me code smells, so I fix while I produce my code. But we have maintenance task to refactor in case we have code smells into our reports	I have sonar lint into my IDE which shows to me code smells, so I fix while I produce my code	Automated detection on produced code	But we have maintenance task to refactor in case we have code smells into our reports	Automated detection on maintained code				

I7	A1: Yeah, there are internal tools that point that stuff out. Got a couple warnings yesterday for duplicate strings, though it was a false alarm in that particular case. Still, I'd rather have false alarms than no alarms at all. A2: For code smell detection? I don't think so. I just noticed I was thinking about a specific tool that runs after you create a code review, but there are also some checks run when we build our java packages that detect vulnerabilities that also can catch some code smells - and that one I'm pretty sure is a public library, but I don't recall which one right now. But no, no external code smell detection tools. A3: In personal projects it would be mostly a matter of lazyness - there's no easier setup than no setup at all, and it is easy to justify to myself that the potential problems aren't worth the hassle. On work on the other hand I'd say it is very important to use code smell detection tools and that it is very much worth the hassle. But to give you an answer, there are concerns against using external static code analysis tools, specially in big tech companies. We do have one on my job, it is just an internal one, and that is why - it is worth developing your own tool if your code is supposed to be that secret.	[...] I just noticed I was thinking about a specific tool that runs after you create a code review	Automated detection during code review	[...] there are also some checks run when we build our java packages that detect vulnerabilities that also can catch some code smells	Automated detection during software build	In personal projects it would be mostly a matter of lazyness - there's no easier setup than no setup at all, and it is easy to justify to myself that the potential problems aren't worth the hassle.	Setup of detection tools is costly in personal projects		