Universidade Federal de São Carlos  
Departamento de Computação

Análise do Projeto Payara por meio de Métricas do SonarQube



Antonio L. Lopes Uliana 628034

Daniel Lucio Masselani de Moura 743525

João Vitor Azevedo Marciano 743554

Juliana Karoline de Sousa 594997

Matheus Kirstus 628310

Disciplina: Engenharia de Software 1

Professor Auri

Setembro de 2018

**Descrição do projeto:**

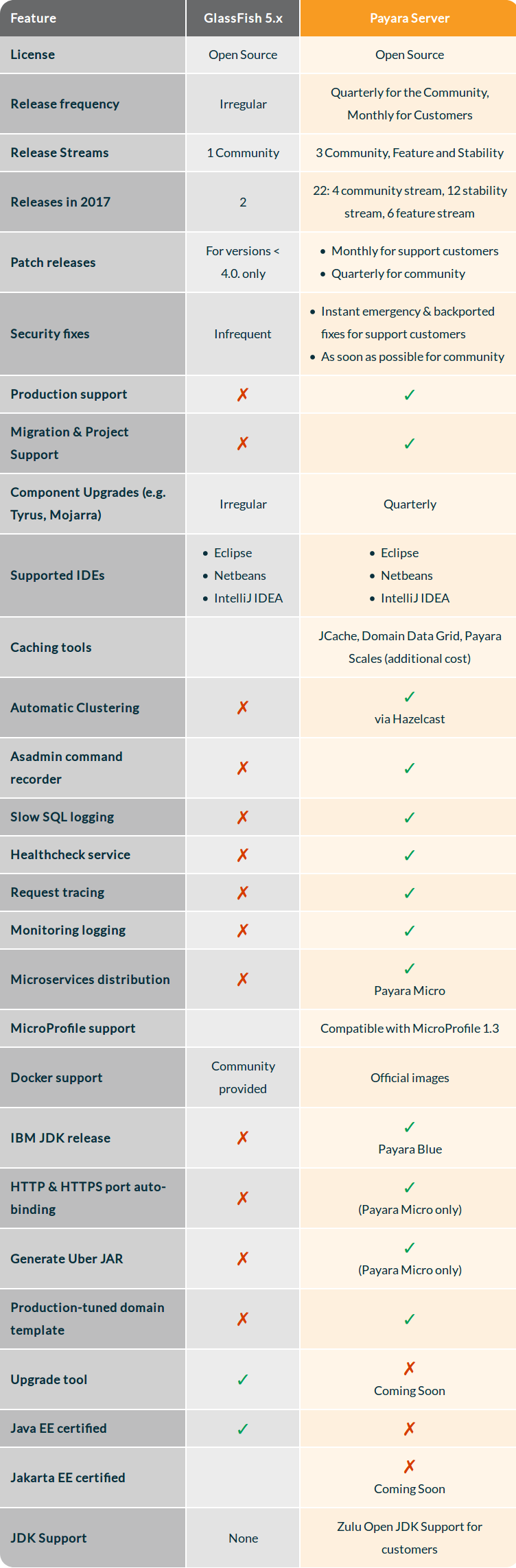
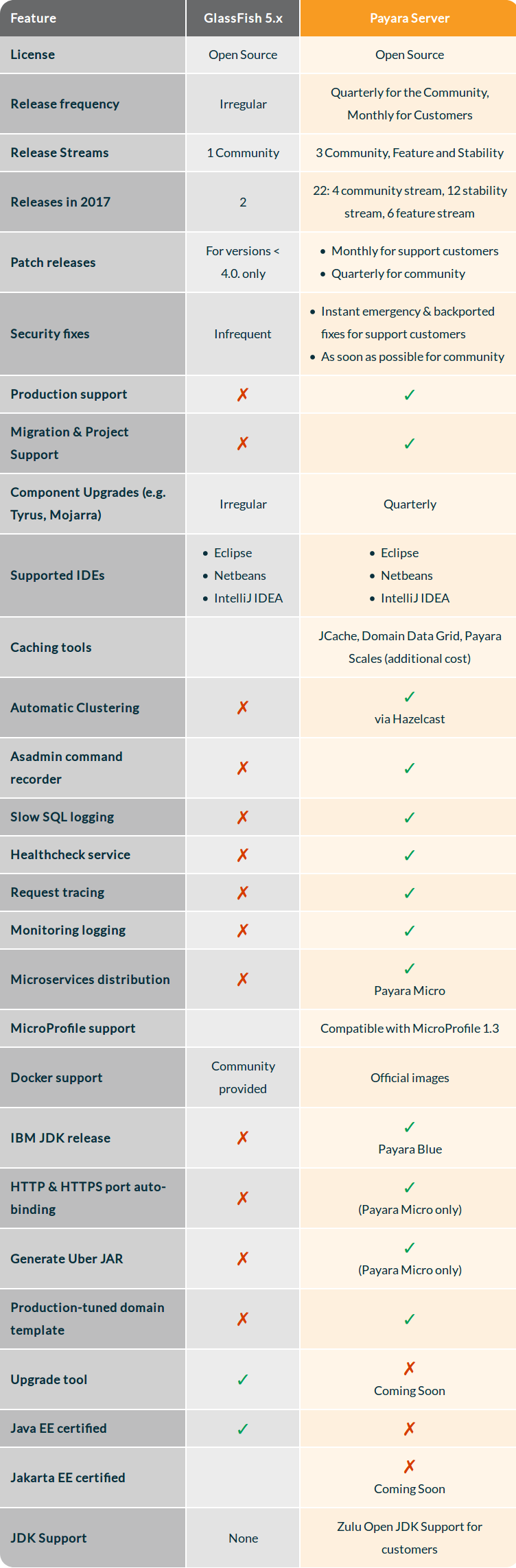
Payara é um servidor de aplicações open source, que surgiu para ocupar a lacuna deixada no mercado, com o fim da Glassfish, oferecendo suporte completo aos serviços prestados por ela. É derivada do Glassfish e foram adicionadas melhorias, e um suporte 24/7.

Serviços oferecidos pelo Payara:

* Payara Server
* Payara Micro
* Payara Support
* Payara Migration & Project Support
* Payara Accelerator Consultancy

Características de destaque:

* Compatível com serviços amplamente utilizados, como Microsoft Azure, Amazon AWS, Google Cloud Platform, etc
* Compatível com Eclipse MicroProfile
* Em conformidade com Jakarta EE
* Otimizado para uso de contêineres, incluindo Docker e Kubernetes.
* A posição do Payara como membro estratégico da comunidade jakarta EE e MicroProfile garante que buscam evoluir os padrões industriais para atender às necessidades dos clientes.
* Mais em: <https://blog.payara.fish/>

Tabela comparativa do Payara com seu predecessor, o GlassFish 5.x

**Histórico da coleta de dados:**

* Primeira coleta de dados: 11/04/2018, 4:33PM
  + Bugs: 1.368
  + Vulnerabilidades: 1.124
  + Code smells: 41.665
* Coleta de dados mais recente: 16/10/2018, 4:32PM
  + Bugs: 1.164
  + Vulnerabilidades: 1.007
  + Code Smells: 39.906

**Quality gate - o que está causando falhas:**

* Linhas duplicadas: 11.8 %
* Grau de confiabilidade: E

O que está causando falhas no código é a porcentagem de linhas duplicadas e o grau de confiabilidade que é muito baixo (E).

**Classificação de bugs e esforço necessário**

* Bugs classificados como “Blocker”: 166
  + Esforço necessário para a correção: 1 dia e 7 horas
* Vulnerabilidades classificadas como “Critical”: 13
  + Esforço necessário para a correção: 2 horas e 25 minutos
* *Code Smells* classificados como “Blocker”: 227
  + Esforço necessário para a correção: 5 dias e 1 hora

**Bugs de maior ocorrência:**

Os bugs do tipo “Resources should be closed” são o de maior ocorrência dentro da categoria ‘Blocker’, somando 154 bugs de um total de 166 bugs dessa categoria.

**Vulnerabilidades de maior ocorrência:**

As concentração das vulnerabilidades críticas estão nas vulnerabilidades do tipo ‘"HttpServletRequest.getRequestedSessionId()" should not be used’, que somam 10 ocorrências, de um total de 13.

***Code Smells* de maior ocorrência:**

Os Code Smells do tipo “Methods and field names should not be the same or differ only by capitalization” possuem um total de 127 ocorrências no projeto, e o total de Code Smells da categoria ‘Blocker’ é de 227.

**Métricas e débito geral:**

O Overall Debt do projeto é de 752 dias. Ou seja, um único programador trabalhando 8 horas por dia, levaria pouco mais de 2 anos para atualizar o projeto, em termos de documentação, correção de code smells, bugs etc. Então, uma equipe de 5 desenvolvedores poderia resolver esses mesmos problemas em 151 dias, por exemplo.

Essa é uma estimativa feita pela SonarQube, que pode ou não refletir a realidade.

***Relato de dados em “Size”***

|  |  |
| --- | --- |
| Quantidade de linhas adicionadas no último commit | 3.109 |
| Linhas de código totais | 792.185 |
| Linhas | 1.596.035 |
| Statements | 317.117 |
| Funções/Procedimentos | 65.280 |
| Classes | 8,314 |
| Arquivos | 7,659 |
| Directories | 1,638 |
| Linhas Comentadas | 228,610 |
| Comentários | 22.4% |

***Outras Métricas Consideradas Relevantes***

|  |  |
| --- | --- |
| Cyclomatic Complexity (num min de casos de teste para uma cobertura de teste completa) | 139476 |
| Cognitive Complexity (quão difícil a aplicação é de se entender - baseado no fluxo de informações da leitura de código) | 141910 |
| False Positive Issues (um caso de teste que indica incorretamente que uma condição particular ou atributo está presente) | 116 |
| Won’t Fix Issues (Trechos críticos de software que são marcados para não serem alterados ou corrigidos, devido a sua criticidade) | 5 |

**Disposição do grupo:**

O grupo estaria disposto a corrigir parte dos problemas detectados. Apesar de muitos, grande parte deles é “fácil” de ser resolvido. Ponteiros nulos, exceções não tratadas, *try’*s sem *finally*, *return*’s duplicados, blocos de código duplicados, métodos com o mesmo nome de parâmetros (com uma letra ou outra em uppercase), etc. Além disso, as estimativas do SonarQube não pareceram muito apuradas, pois foi aberto trechos de código em que a ferramenta estimou tempo de resolução de 30 minutos, e obteve-se a impressão de que poderia ser resolvido com muito menos.