**Herramientas Seleccionadas**

**Sistema de Diagnóstico y Agendamiento de Citas (SDAC)**

**Versión 1**

**Jairo Daniel Bautista Castro**

**Miguel de Oliveira Dias Gonçalves**

**Herramientas de control de versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterio** | **Peso** | **BitBucket** | **GitHub** | **GitLab** |
| Comunidad de usuarios. | 30 | 7 | 10 | 6 |
| Precios | 40 | 8 | 7 | 8 |
| Integración | 30 | 9 | 8 | 7 |
| TOTAL | | 80 | 82 | 71 |

La herramienta Github es la elegida para el control de versiones en vez de Bitbucket y GitLab, pues se tiene en cuenta la gran comunidad de usuarios que se tiene, además de que con esto se espera que los desarrolladores para cada proyecto estén acostumbrados a esta herramienta y la curva de aprendizaje sea por lo tanto más rápida para el equipo.

**Herramientas de compilación**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterio** | **Peso** | **Maven** | **Gradle** | **Ant** |
| Integración con IDEs | 25 | 9 | 5 | 8 |
| Lenguajes soportadas | 35 | 3 | 8 | 3 |
| Facilidad de utilización | 40 | 8 | 7 | 5 |
| TOTAL | | 65 | 68,5 | 50,5 |

La herramienta de compilación elegida es Gradle, pese a que Maven es más utilizado y tiene una mayor integración con diferentes IDEs de desarrollo. Gradle es una herramienta más expresiva y soporta más lenguajes de programación, un factor clave para nuestro proyecto pues se van a utilizar varios lenguajes.

**Herramientas de automatización de pruebas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterio** | **Peso** | **Junit** | **TestNG** | **Parasoft Test** |
| Más Usado | 30 | 10 | 7 | 6 |
| Facilidad de uso | 30 | 8 | 8 | 7 |
| Reportes de código | 40 | 7 | 8 | 6 |
| TOTAL | | 82 | 77 | 63 |

La herramienta que se escoge para las pruebas unitarias es Junit, ya que es el estándar de pruebas unitarias de Java. También se lo selecciona por su usabilidad y versatilidad para el desarrollo de pruebas unitarias.

**Herramientas de integración continua**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterio** | **Peso** | **Jenkins** | **Travis CI** | **Circle CI** |
| Lenguajes soportadas | 30 | 8 | 10 | 7 |
| Continuous Deployment | 40 | 9 | 8 | 8 |
| Precio | 30 | 10 | 4 | 5 |
| TOTAL | | 90 | 74 | 68 |

Para la herramienta de integración continua se escogerá Jenkins, pues para la necesidad del proyecto de ser On premise, es una mejor solución que Travis CI o Circle CI. Además el precio, la versatilidad a la hora de configuración con la instalación de plugins y su facilidad de integración con Java fueron los factores claves a la hora de escoger esta herramienta.

**Herramientas de seguimiento de errores/defectos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterio** | **Peso** | **Mantis** | **Jira** | **Bugzilla** |
| Facilidad de uso | 30 | 6 | 7 | 8 |
| Precio | 40 | 7 | 5 | 8 |
| Reporte de errores | 30 | 7 | 6 | 8 |
| TOTAL | | 67 | 59 | 80 |

Para la herramienta de bugs se escoge Bugzilla, ya que es una herramienta libre y gratis de utilizar, y además sus reportes de errores son más completos que los de los competidores. Por otro lado su sistema de rastreo de errores en tiempo real nos ayudará a rastrear y corregir los errores más fácilmente que las otras dos herramientas de seguimiento de errores.