|  |  |
| --- | --- |
| عنوان پروژه | زیرساخت تولید محصولات نرم افزاری |
| عنوان گزارش | **بررسی کیفیت پیاده سازی کد منبع – نسخه ارائه شده اولیه** |
| موضوعات مورد پوشش ذیل عنوان | Clean Code , Code convention , Design Patterns , SOLID principles , LLD |
| واحد تنظیم کننده | زیرساخت و یکپارچه سازی |
| نام تنظیم کننده | رضا اخگری |

|  |  |
| --- | --- |
|  | **الف – پیش فرضها** |
|  | *تبصره: در صورتی که مفروضات از اساس نادرست باشد ، مطرح شدن گزارش جاری موضوعیت خود را از دست داده و باید ابتدا به آنها پرداخته شود.* |
|  | * فاز شناخت و مطالعات اولیه به درستی طی شده و درک درستی از نیازمندیهای بهره بردار (در اینجا بطور اختصاصی تیم(های) توسعه محصولات)در سطح اولیه صورت پذیرفته است؛ |
|  | * سند معماری در سطح کلان تهیه ، تصویب شده و قابل ارجاع است . * انتخاب فناوری های موجود قابل استفاده در سطوح مختلف انجام پذیرفته و آنچه که می بایست بصورت اختصاصی جهت پیشبرد پروژه بازتولید یا اختصاصی سازی شود، به روشنی مستند شده است؛ |
|  | * پروژه سطحی از انتزاع (abstraction) را در هدف گذاری خود تعریف کرده است که قابلیت تجاری سازی و تعمیم موضوعی در سایر پروژه های خارج از مجموعه اما مرتبط را داشته باشد. |

|  |  |
| --- | --- |
|  | **ب – چرا این گزارش تنظیم شده است؟!** |
|  | * نوشتن کد یک مهارت و هر مهارتی بر اساس برخی شاخصها قابل اندازه گیری است. کدها همواره در چرخه ریفکتورینگ بهبود پیدا می کند. |
|  | * تحویل کد برای اولین بار از سوی توسعه گر منجر به بدهی فنی می شود. اگر این بدهی بازپرداخت نشود هر لحظه که از عمر این کد که کاملا درست نوشته نشده است بگذرد بهره‌ای به بدهی قبلی افزوده می‌شود. بسیاری از تیم ها زیر فشار بدهی ناشی از بی‌توجهی به این‌گونه پیاده‌سازی‌های بازنویسی‌نشده ممکن است زمین‌گیر شوند و پیشرفت آنها متوقف شود. * باید به درستی تشخیص داده شود که منشا بدهی فنی ناشی از چیست: بی‌ملاحظه بودن (reckless) ، مصلحت اندیشی (prudent) ، تعمد (deliberate) و یا عمل سهوی (inadvertent). |
|  | * اگر تصور بر این است که برخی ابزارهای ریفکتورنیگ مانند SonarLint و یا SonarQube پاسخگوی تمامی نیازهای این حوزه است سخت در اشتباه هستید چون تجربه و مهارت ریفکتورنیگ توسط توسعه گران قابل جایگزینی با موارد نام برده شده نیست. آنها تنها تسهیل گر هستند. * قانون پیشاهنگی: همیشه وقتی اردوگاه را ترک می‌کنید، تمیزتر از زمان ورودتان باشد. اگر زباله‌ای روی زمین دیدید، بدون آن که دنبال این باشید که چه کسی آن را ریخته است، زباله را بردارید و آن جا را تمیز کنید. با این کار، آگاهانه محیط را برای گروه بعدی آماده‌تر می‌کنید. با اجرای این قانون همواره تلاش می‌کنیم تا هر بار که به سراغ کد میرویم، طراحی و پیاده سازی آن را کمی بهتر از گذشته کنیم و نه بدتر. |

|  |  |  |
| --- | --- | --- |
|  | **ج - نگاه کلی به آنچه هم اکنون در دسترس است:** |  |
|  | نقاط قوت | نقاط ضعف |
|  | * درک صحیح از مسئله در پیاده سازی مشهود است | * توسعه قائم به فرد / تولید کد خصوصی - عدم وجود انگاشت های طراحی در پیاده سازی (LLD) * عدم وجود کامنت و با هرگونه راهنمای اولیه برای فهم کد توسط غیر |
|  | * آشنایی اصولی با Java core و استفاده قابل قبول از ساختارهای موجود خصوصا مباحث پایه ای OOP | * عدم استفاده از الگوهای پیاده سازی و اصرار بر رویکردهای قابل دسترس جهت حصول به نتیجه در کوتاه ترین زمان ممکن در برخی قسمت ها (*مطالعه موردی در ادامه گزارش*) |
|  | * پیاده سازی انتزاعی مناسب برخی از ساختارها با رویکرد توسعه اختصاصی برای بهره بردار | * نقض برخی از اصول SOLID و به صورت مشخص OCP |
|  |  | * Lazy coding در برخی قسمت ها و عدم توجه به Code convention |
|  |  | * پیکیجینگ کلاسها و سایر ساختارها با مشکل جدی روبروست. عدم انتخاب صحیح عناوین artifact ها مبتنی بر اصول کلی Naming convention for Maven Artifacts |
|  |  | * به نظر می رسد توسعه گر(ان) نسبت به فریمورک اسپرینگ حداقل بر پایه کدهای تولید شده تا زمان تنظیم این گزارش از تسلط و یا اشراف جامعی برخوردار نیست و یا به دلیل مشکلات زمان بندی نتوانسته نسبت به کدها و ساختارهای پیشنهادی خود تجدید نظر کند (*مطالعه موردی در ادامه گزارش*) |
|  |  |  |
|  |  |  |
|  |  |  |

جدول بررسی اعلام مشکلات کد

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | وجود مشکل در سطح پیاده سازی | | | | | | | | | | | شاخص بررسی |  |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  | Long Parameters | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  | Long Method | 2 |
|  |  |  |  |  |  |  |  |  |  |  |  | Contrived Complexity | 3 |
|  |  |  |  |  |  |  |  |  |  |  |  | Primitive Obsession | 4 |
|  |  |  |  |  |  |  |  |  |  |  |  | Data Clumps | 5 |
|  |  |  |  |  |  |  |  |  |  |  |  | God Class | 6 |
|  |  |  |  |  |  |  |  |  |  |  |  | Conditional Complexity | 7 |
|  |  |  |  |  |  |  |  |  |  |  |  | Refused Bequest | 8 |
|  |  |  |  |  |  |  |  |  |  |  |  | Temporary field | 9 |
|  |  |  |  |  |  |  |  |  |  |  |  | Alternative classes with different interfaces | 10 |
|  |  |  |  |  |  |  |  |  |  |  |  | Divergent change | 11 |
|  |  |  |  |  |  |  |  |  |  |  |  | Solution sprawl | 12 |
|  |  |  |  |  |  |  |  |  |  |  |  | Shotgun Surgery | 13 |
|  |  |  |  |  |  |  |  |  |  |  |  | Parallel interface hierarchies | 14 |
|  |  |  |  |  |  |  |  |  |  |  |  | Feature Envy | 15 |
|  |  |  |  |  |  |  |  |  |  |  |  | Inappropriate Intimacy | 16 |
|  |  |  |  |  |  |  |  |  |  |  |  | Excessive Exposure | 17 |
|  |  |  |  |  |  |  |  |  |  |  |  | Message Chain | 18 |
|  |  |  |  |  |  |  |  |  |  |  |  | Middle Man | 19 |
|  |  |  |  |  |  |  |  |  |  |  |  | Comments | 20 |
|  |  |  |  |  |  |  |  |  |  |  |  | Dead Code | 21 |
|  |  |  |  |  |  |  |  |  |  |  |  | Duplicate Code | 22 |
|  |  |  |  |  |  |  |  |  |  |  |  | Speculative Generality | 23 |
|  |  |  |  |  |  |  |  |  |  |  |  | Lazy Class | 24 |
|  |  |  |  |  |  |  |  |  |  |  |  | Data Class | 25 |