**Conception, maintenance et sécurité**

# Principes de base

* Keep it Simple Stupid (KISS)
* You Ain't gonna need it (YAGNI)
* Don't repeat yourself (DRY)
* Pensez SOLID
* Command-query separation (CQS)

# Conception de classes

* 3.1  Les méthodes de services web doivent retourner une instance de classe implémentant *IParamSorti*
* 3.2  Ordre de déclaration des membres d’une classe
* 3.3  Déclarez une seule variable par ligne
* 3.4  Mettre une seule classe ou interface par fichier
* 3.5  Ne pas modifier les interfaces publiées

# Types et instructions

* 4.1  L’initialisation de propriétés d’une instance ne doit pas être faite dans un ordre prédéterminé
* 4.2  Ne pas utiliser de propriétés mutuellement exclusives
* 4.3  Retournez un *IEnumerable<T>* ou un *ICollection<T>* plutôt qu'une liste concrète
* 4.4  Les propriétés, méthodes et paramètres représentant des chaines ou des collections ne doivent jamais retourner *null*
* 4.5  Évitez l’utilisation de *null* pour les types complexes
* 4.6  Les paramètres, propriétés et variables doivent être le plus spécifique possible

# Conception générale

* 5.1  Lors d'une erreur technique, lancez une exception plutôt que de retourner une valeur d'état
* 5.2  Utilisez des exceptions pour gérer les erreurs et non pour contrôler le flot d’exécution
* 5.3  Privilégiez *IEnumerable<T>* et *yield* pour le retour de collections coûteuses en ressources

# Utilisation du framework

* 6.1  Utilisez les alias de types du langage plutôt que les types de l’espace-nom *System*
* 6.2  Privilégiez l’utilisation des appels asynchrones (*async* et *await*)
* 6.3  Évitez l’utilisation des signatures de méthode "Async Void"

# Maintenance

* 7.1  Attrapez des exceptions spécifiques
* 7.2  Évitez les nombres et les chaines magiques
* 7.3  Par défaut, mettre privés ou internes tous les membres et méthodes d'une classe
* 7.4  Évitez les conditions avec des double-négatifs
* 7.5  Déclarez et initialisez les variables le plus près possibles de la première utilisation
* 7.6  Toujours utiliser des blocs {} avec les identifiants *if*, *else*, *while*, *for*, *foreach* et *case*
* 7.7  Préférez une simple affectation plutôt qu’une déclaration *if-else*
* 7.8  Ne pas utiliser de paramètres *ref* ou *out*
* 7.9  Pour relancer une exception, utilisez *throw*
* 7.10  Limitez les lambdas imbriquées
* 7.11  Utilisez l’inférence de type lorsqu’elle améliore la lisibilité
* 7.12  Privilégiez les initialiseurs d’objets et de collection plutôt que des instructions séparées
* 7.13  Utilisez le mot clé *nameof*
* 7.14  Implémentez *IDisposable* lorsque nécessaire
* 7.15  Privilégiez les collections génériques aux collections non typées
* 7.16  Ne pas utiliser de variables statiques
* 7.17  Interdit d’utiliser l’instruction « *Goto* »
* 7.18  Interdit d’utiliser l’instruction « *GC.Collect* »

# Code smell

Les éléments suivant peuvent indiquer des lacunes dans le code

* 12.1  Éliminez le code dupliqué
* 12.2  Une énumération ne doit pas définir le comportement
* 12.3  Ne pas cacher les exceptions
* 12.4  Une classe de base ne peut avoir une dépendance vers ses dérivées
* 12.5  Évitez les commentaires inutiles ou inconsistants
* 12.6  Le comportement attendu doit être celui implémenté
* 12.7  Évitez les méthodes avec plus de 5 paramètres
* 12.8  Effacez le code mis en commentaire
* 12.9  Une méthode ne devrait pas avoir plus de 200 lignes
* 12.10  Une classe ne devrait pas avoir plus de 2000 lignes
* 12.11  Éliminez les avertissements de compilation
* 12.12  Ne pas cacher les avertissements de compilation avec des directives *#pragma*

# Sécurité applicative

* 10.1  Entreposez les données sensibles ou nominatives dans des dépôts sécuritaires
* 10.2  Les erreurs techniques ou fonctionnelles ne doivent pas contenir de données sensibles
* 10.3  Validez les intrants
* 10.4  Encodez les sortants
* 10.5  Encryptez les données nominatives ou privées
* 10.6  Limitez la conservation d’état dans les cookies
* 10.7  Sécurisez les cookies
* 10.8  Interdit d’appeler *Server.Execute*
* 10.9  Limitez l’utilisation de *XPath*
* 10.10  Désactivez l’*autocomplete* pour les champs avec valeur sensible ou nominative (HTML)
* 10.11  Terminez l’exécution lors d’une redirection ASP.NET
* 10.12  Ne pas mettre de valeur de clé primaire dans les champs cachés (HTML)

**Nomenclature et apparence**

|  |  |
| --- | --- |
| **PascalCase** | |
| Espace-nom | RAMQ.PPP.NomSignificatif |
| Classe | ContratsDispensateur |
| Interface | IContratsDispensateur |
| Méthode | ObtenirContratsDispensateur |
| Propriété | NomDispensateur |
| Évènement | ContratTermine |
| Variable de classe publique, interne ou protégée | NomDispensateur |
| Constante | MontantMaximum |
| Énumération | Environnement |
| Paramètre de type générique | T, TInfoDispensateur |
| Contrôle graphique publique, interne ou protégée | NomDispensateur |
| **camelCase** | |
| Variable de classe privée | \_nomDispensateur |
| Variable locale | nomDispensateur |
| Paramètre d’une méthode | nomDispensateur |
| Contrôle graphique privée | nomDispensateur |

# Nomenclature

* 2.1  Utilisez le français
* 2.2  Utilisez la bonne capitalisation selon l’élément
* 2.3  Évitez les nombres dans les identifiants
* 2.4  Ne pas préfixer les variables ou paramètres
* 2.5  S’il est nécessaire de raccourcir un identifiant, utilisez les mnémoniques RAMQ
* 2.6  Nommez les identifiants selon leur utilité et non selon leur type
* 2.7  Ne pas répéter le nom de la classe ou de l'énumération dans ses membres
* 2.8  Bien nommer les propriétés
* 2.9  Le nom des méthodes doivent avoir un verbe puis un nom
* 2.10  Utilisez un souligné pour les paramètres inutilisés d’une lambda
* 2.11  Le nom d’un paramètre d'une fonction lambda, d’une requête LINQ ou une variable servant d’index dans une boucle peut être d’une seule lettre

# Style

* 9.1  Utilisez le bon jugement pour la longueur des lignes
* 9.2  Ne pas changer l’indentation par défaut de Visual Studio
* 9.3  Toujours mettre des espaces blancs autour des opérateurs
* 9.4  La longueur des identifiants est illimitée

# Documentation

* 11.1  Les commentaires doivent être en français
* 11.2  Tous les éléments publics, protégés ou internes doivent être commentés
* 11.3  Évitez les commentaires « *inline* »
* 11.4  Commentez le code seulement lors d’algorithmes ou de décisions complexes
* 11.5  Ne pas mettre d’historique de modifications dans les commentaires.

# Développement web

* 8.1  Utilisez le Module Pattern (javascript)
* 8.2  Utilisez le docType HTML5