## Изисквания към качеството на системи в рамките на електронното управление

# Съдържание

1. [Документация на интерфейсите](#1-Документация-на-интерфейсите)
2. [Сигурност](#2-Сигурност)
3. [SDK](#3-SDK)
4. [Производителност](#4-Производителност)
5. [Отказоустойчивост](#5-Отказоустойчивост)
6. [Браузъри и уеб](#6-Браузъри-и-уеб)
7. [Качество на софтуерната разработка](#7-Качество-на-софтуерната-разработка)

Този документ описва изискванията към различни аспекти на качеството на изгражданите софтуерни системи.

### 1. **Документация на интерфейсите**

Всеки предоставен REST приложно-програмен интерфейс трябва да бъде  
документиран чрез API Blueprint  
(<https://github.com/apiaryio/api-blueprint>) или сходна технология.  
  
Аналогично представяне е нужно и за SOAP интерфейсите.

### 2. **Сигурност**

Следващите изисквания подлежат на промяна с цел да бъдат отразени  
последните открития в сферата на информационната сигурност.  
  
Имплементирането на промени в софтуерните системи в отговор на промените  
в тези изисквания, които са в резултат на откритие на проблем със  
сигурността, който може да доведе до изтичане на данни, трябва да се  
случва незабавно.  
  
1. \*\*Криптографски протоколи\*\*  
  
 Комуникацията между интерфейсите трябва да се извършва през HTTPS  
 протокол, с използване на минимум TLS 1.2 за сигурна автентикация и  
 криптиране на канала.  
  
 Използването на доказано дефектни протоколи като SSL 1.0/2.0/3.0 е  
 недопустимо.  
  
 При изграждане на нови проекти, следва да се използват последните версии  
 на протоколите и алгоритмите.  
  
 При надграждане на съществуващите системи, където сървърите и средата не  
 може да бъде приведена до последна версия, трябва да се използват  
 максималните поддържани версии.  
  
2. \*\*Комплекти от шифри\*\*  
  
 Допустимите Cipher Suites за TLS 1.2 (ограничено подмножество, което  
 вероятно ще влезе в TLS 1.3)  
  
 \*\*Key exchange / agreement\*\*  
  
 Ephemeral Diffie-Hellman (TLS\\_DHE), parameters \>3072 bit (не се  
 поддържат преди Java 1.7)  
  
 Elliptic Curve Ephemeral Diffie-Hellman (TLS\\_ECDHE) с елиптична крива с  
 поне 256-бита големина на частния ключ, тъй като поддържат “Perfect  
 Forward Secrecy”  
  
 \*\*Authentication\*\*  
  
 RSA с частен ключ с дължина поне 3072 бита(\>=3072bit), ECDSA с  
 елиптична крива с частен ключ с дължина поне 256 бита.  
  
 \*\*Bulk ciphers\*\*  
  
 AES-128, AES-256 в режим GCM или Galois/Counter Mode или друг  
 authenticated encryption режим,.  
  
 \*\*Hash-based Message Authentication Code\*\*  
  
 SHA-256, SHA-384, SHA-512  
  
 \*\*Позволени комбинации:\*\*  
  
 TLS\\_DHE\\_RSA\\_AES128\\_GCM\\_SHA256   
 TLS\\_DHE\\_RSA\\_AES256\\_GCM\\_SHA384   
 TLS\\_ECDHE\\_RSA\\_AES128\\_GCM\\_SHA256   
 TLS\\_ECDHE\\_RSA\\_AES256\\_GCM\\_SHA384   
 TLS\\_ECDHE\\_ECDSA\\_AES128\\_GCM\\_SHA256   
 TLS\\_ECDHE\\_ECDSA\\_AES256\\_GCM\\_SHA384   
  
3. \*\*Уеб сигурност\*\*  
   
 Изградените системи трябва да спазват добрите практики при защита от  
 уеб-атаки като XSS, CSRF, SQLi и др.  
 Минимумът от изисквания, които трябва да бъдат покрити, е описан от  
 OWASP в <a href="https://www.owasp.org/index.php/OWASP\_Top\_Ten\_Cheat\_Sheet">Top Ten Cheat Sheet</a>  
   
4. \*\*Електронно подписване\*\*  
  
 Когато е нужно електронно подписване, се прилага хеш функция върху  
 документ, който се подписва, и резултатът от хеш функцията се подава към  
 сигурното устройство (HSM/смарткарта) за подписване.  
  
 Електронното подписване става с използването на електронен печат на  
 организацията. Този електронен печат следва да бъде одобрен от орган,  
 който регулира издаването и използването им (CA към централния  
 регистър). Електронните печати следва да могат да се използват от  
 софтуера, който реализира услугите, без човешка намеса.  
  
 Когато потребител трябва да подпише документ, браузърите не предоставят  
 стандартна функционалност, което води до нуждата от специфични и  
 непреносими в различни браузъри решения. Тъй като всеки притежател на  
 квалифициран електронен подпис разполага с desktop софтуер за електронно  
 подписване, следва да се обмисли опция за качване на вече подписани  
 файлове.  
  
5. \*\*Управление на пароли\*\*  
  
 Паролите, когато се такива се използват в АИС-а, следва да бъдат  
 съхранявани в базата данни след използване на алгоритми като bcrypt,  
 scrypt или PBKDF2. Следва да имат и изискване за дължина, поне 6  
 символа, и препоръка за използване на изрази, вместо единични думи.

### 3. **SDK**

Всеки АИС (вкл. регистрите) трябва да предоставя SDK-и за достъп до  
уеб-услугите си на езика, на който е разработен АИС-а. При разработване  
на АИС-и-консуматори на друг програмен език, те предоставят за публично  
ползване SDK-и, които са написали.  
  
Базови SDK, включващи компоненти за достъп до стандартизирания процес по  
заявяване на данни (описан в т.6) ще бъдат налични от централния  
координатор, а специфичните SDK-и за всеки АИС трябва да ги надграждат.  
  
При идентифициране на функционалност, необходима в множество проекти,   
тя трябва да бъде отделена в отделна библиотека (SDK).  
  
SDK-ите трябва да бъдат публикувани в стандартните хранилища за  
библиотеки (напр. Maven Central, NuGet, PEAR), като параметрите за  
тяхното включване в други проекти трябва да бъдат дефиниране в README  
файл в хранилището на проекта.

### 4. **Производителност**

- Всяка система трябва да поддържа т.нар. rate-limiting -  
 механизъм, ограничаващ всеки заявител до не повече от определен  
 брой заявки в секунда. Примерен отговор при достигане на лимита  
 изглежда така:  
  
```  
429 Too Many Requests  
Content-Type: application/json  
Rate-Limit: 200.00  
{  
 "error": {  
 "description":"The request rate limit of 200.0 requests/s is exceeded."  
 }  
}  
```  
  
- кеш - следва да се използват различни видове кешове, с цел избягване  
 на излишно четене от диск или от мрежов ресурс. Основен кандидат за  
 кеширане е достъпът до базата данни. Системата за кеширане може да  
 бъде локална (в текущата виртуална машина/процес) или разпределена.  
 Някои разпределени системи за кеш са: Redis, Infinispan, EhCache,  
 JBoss DataGrid, Memcached, AppFabric Caching, Hazelcast...  
 Кои елементи/завки/извиквания да бъдат кеширани зависи от анализ на   
 тяхното използване. Кешът може да има негативен ефект, в зависимост   
 от hit-to-miss съотношението.   
 Препоръчваната политика за изтриване на елементи от кеша   
 при препълване на LRU (least recently used)  
 Регистрите не трябва да предоставят остарели данни. При промяна на   
 даден запис, съответният елемент в кеша трябва да става невалиден.  
  
  
- опашки - системите могат да обработват заявки чрез опашки, например  
 за заявки, очакващи асинхронен (ASYNC) отговор. Заб: заявка за  
 обиРегистрите не трябва да предоставят остарели данни. При промяна  
 на даден запис, съответният елемент в кеша трябва да става  
 невалиден. Обикновено четене от базата данни по принцип не би следвало да има  
 нужда от опашка.  
  
- времето за отговор на всяка заявка за данни към първичен  
 регистър трябва да е не повече от 200 милисекунди при стандартно  
 натоварване.

### 5. **Отказоустойчивост**

Всеки АИС трябва да е напълно функционален дори при нарушение на части  
от софтуерната, хардуерната и мрежовата инфраструктура:  
  
- приложните сървъри трябва да бъдат в клъстер и да са поне два.  
 Препоръчително е в самите сървъри да няма съдържание (т.е. да бъдат  
 stateless), което да позволи хоризонтално мащабиране.  
  
- трябва да се използва т.нар. load balancer за балансиране на  
 натоварването към приложните сървъри.  
  
- сървърите за данни трябва или да бъдат в клъстер (ако системата за  
 управление на бази данни го позволява), или да поддържат т.нар.  
 failover към друг сървър, към който основният се репликира.  
  
- Ако даден АИС (вкл. първичен регистър) не отговори успешно, заявките  
 се маркират като “незавършени” в системата на заявителя и се  
 изпълняват отново след време.

### 6. **Браузъри и уеб**

Интернет-приложенията трябва да работят и изглеждат по еднакъв начин на  
последните две версии на браузъри с над 10% пазарен дял. Не се допуска  
някоя система да работи “само на Internet Explorer”, например.  
  
Не трябва да се разчита на допълнителни браузърни приставки или разширения  
от страна на клиента за тяхното нормално функциониране, с изключение на   
функционалността по електронно подписване. За електронно подписване се   
препоръчва използването на <a href="http://open-eid.github.io/">естонския подход</a>.  
  
Публичният интерфейс на всяка система трябва да бъде удобен за ползване  
чрез мобилни устройства, като за целта е желателно интерфейса автоматично  
да се оразмерява, в зависимост от размера на екрана (responsive).   
Създаването на напълно отделна мобилна версия на интерфейса е нежелателно.  
  
Публичната част на всяка система трябва да има валиден цифров  
сертификат, издаден от разпознато в браузърите Certificate Authority.

### 7. **Качество на софтуерната разработка**

Разработката на софтуерните решения трябва да следва съвременни добри практики. Това включва:  
  
- Документиране на компонентите в изходния код  
- Покритие на минимум 50% от изходния код с тестове  
- Възможност за инсталация с една команда  
- Използване на т.нар. Continuous integration  
- Използване на т.нар. Dependency management