**1. HTTP REST Best practices or conventions**

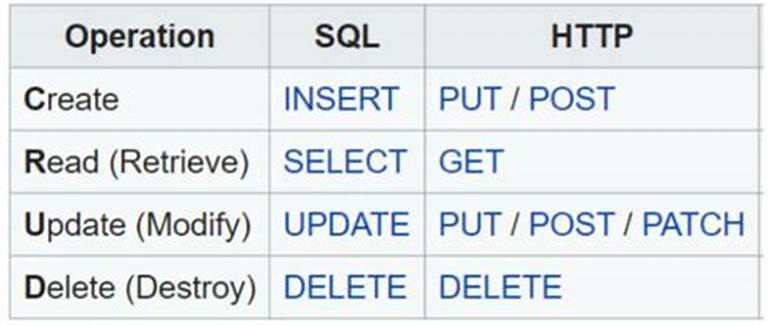
**استخدام الأساليب الصحيحة (HTTP Methods):**

يعني مثلاً لو عايز تجيب حاجة من السيرفر، استخدم GET زي لما بتشتري عصير من البقالة.

لو عايز تبعت بيانات عشان تضيف حاجة جديدة، استخدم POST زي لما بتكتب طلبية في المطعم.

لو عايز تحدث حاجة موجودة بالفعل، استخدم PUT زي لما بتغير طلبية محددة من المطعم.

ولو عايز تمسح حاجة موجودة، استخدم DELETE زي لما بتمسح الطلبية لو عدت عليها.



**التحكم في الردود (Response Codes):**

لما تبعت طلب، السيرفر بيرد عليك بكود يبين إذا كان الطلب ناجح أو فيه مشكلة.

زي لما تشوف لوحة تحكم في السيارة، لو كانت الأضواء خضرا بتبقى طلبك تمام، لو كانت حمرا فيه حاجة غلط.

**توثيق جيد (Good Documentation):**

يعني لازم تكون واضح وعندك توثيق كويس لكل الحاجات اللي بتقدمها من السيرفر.

زي لما بتبقى بتعمل دليل استخدام لجهاز جديد عشان الناس يفهموا ازاي يستخدموه.

**استخدام رموز الخطأ بشكل مناسب (Error Handling):**

يبقى لو حصل أي مشكلة في الطلب، السيرفر يرجعلك كود يوضحلك المشكلة.

زي لما تسأل الجار عن عنوان وهو يقولك مش عارف، يبقى في مشكلة في المعلومة اللي طلبتها.

**استخدام روابط (Links):**

يعني لما تجيب حاجة، حط رابط يودي لحاجات مرتبطة بيها.

زي لما بتخلي الصفحة مليانة بروابط لصفحات تانية علشان يسهل على الناس يتصفحوا الموقع.

التحقق من الهوية والأمان (Authentication and Security):

اتأكد إن اللي بيستخدم السيرفر هو نفسه اللي مسجل بالطريقة الصحيحة.

زي لما بتتأكد إن اللي بتكلمه في الهاتف هو الشخص اللي بتتواصل معاه بالفعل.

ا**ستخدام النماذج (Data Models):**

يبقى لازم تستخدم تنسيق البيانات اللي يسهل فهمها للجميع، زي JSON.

زي لما بتكتب رسالة عشان توصل معلومة لحد، تحاول تكون الرسالة سهلة ومفهومة.

**2. SQL injection**

يا عم، الـ SQL injection ديه مشكلة كبيرة في عالم تطوير الويب، وهي بتحصل لما تسمح لحد يدخل بيانات في تطبيقك بشكل غير آمن، وده بيسمح لهم يعملوا عمليات SQL في قاعدة البيانات بدون إذن. فكلمة "injection" بتعني تحقين أو حقن، يعني بيتم حقن الأوامر في الـ SQL.

مثلاً، لو كان عندك صفحة لتسجيل الدخول، والكود الخلفي بيشتغل بهذا الشكل:

SELECT \* FROM users WHERE username = '$username' AND password = '$password';

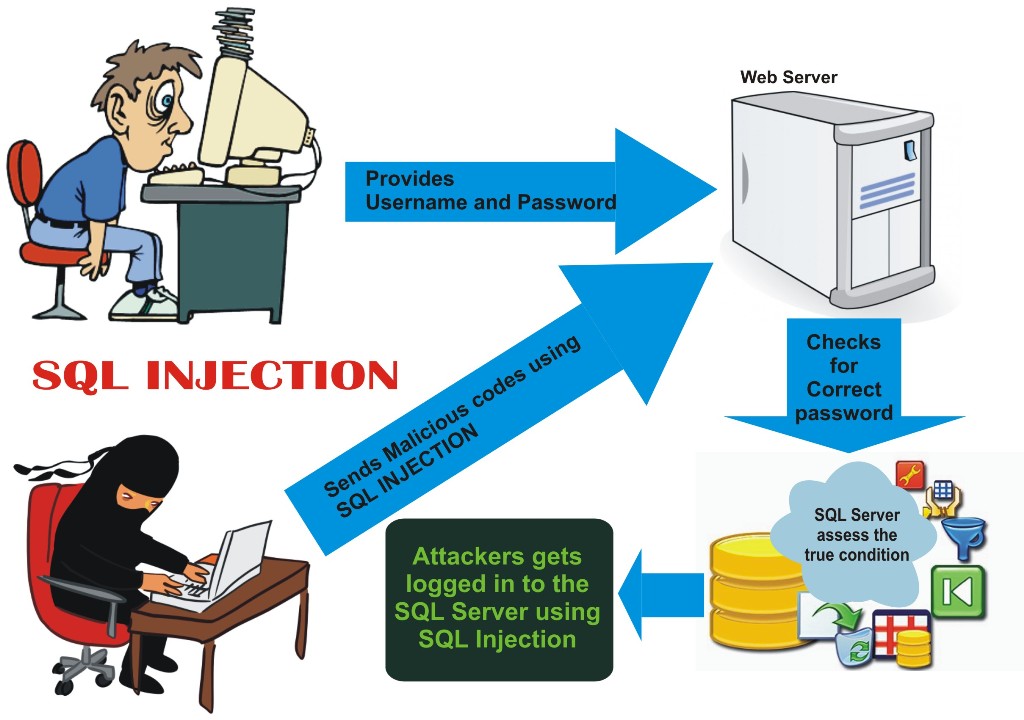
وبيتم استخدام قيمة المتغيرات `$username` و `$password` بدون تصفية، فيمكن لشخص ما أن يدخل قيمة مشبوهة مثل:

$username = 'admin';

$password = ' OR '1'='1

وبما أن الشرط بعد كلمة السر "OR" دائمًا يتحقق (لأنه دائمًا يكون "1=1")، سيعيد الاستعلام جميع الصفوف في جدول المستخدمين، وبالتالي يمكن للشخص الوصول لحساب المدير دون معرفة كلمة المرور الصحيحة.

لحماية التطبيق الخاص بك من هذا النوع من الهجمات، يجب استخدام تعبيرات معقدة ومتقدمة لتصفية البيانات المدخلة وتجنب استخدام البيانات المدخلة مباشرة في الاستعلامات SQL. يمكن استخدام تقنيات مثل استخدام بيانات معلمة مع الاستعلامات المُعدة مسبقًا أو استخدام ظواهر متقدمة مثل قوائم البيضاء للمدخلات.



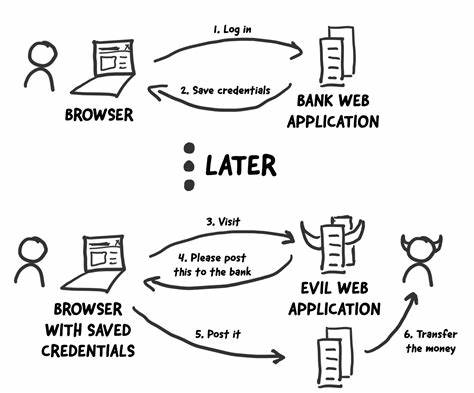
**3. CSRF**

طيب، الـ CSRF ديه اختصار لـ "Cross-Site Request Forgery"، وده نوع من أنواع الهجمات الأمنية على تطبيقات الويب. في هذا النوع من الهجمات، يتم استغلال ثقة المستخدم في المواقع التي يكون لديهم جلسة مفتوحة.

بمعنى آخر، الهجوم يستفيد من الحقيقة أن المستخدم مسجل الدخول في موقع ما ولديه جلسة نشطة، فيقوم بتشغيل أوامر معينة (كالنقر على رابط) دون علمه. على سبيل المثال، يمكن للمهاجم إنشاء صفحة ويب تحتوي على استدعاء غير مرئي إلى موقع الضحية، وعندما يزور الضحية هذه الصفحة وهو متصل بحسابه، يتم تنفيذ الطلب بنجاح.

على سبيل المثال، يمكن للهاكر أن يرسل رابطاً إلى الضحية يحتوي على طلب معين، مثل تغيير كلمة المرور، وعندما ينقر الضحية على الرابط دون أن يشعر، يتم تنفيذ الطلب بالفعل. هذا يمكن أن يكون خطيرًا جدًا إذا كان الضحية مسجلة في مواقع مهمة مثل البنوك أو الشبكات الاجتماعية.

لحماية التطبيقات من هذا النوع من الهجمات، يجب اتخاذ إجراءات أمان مثل استخدام علامات الطلبات المتزامنة (CSRF tokens)، حيث يتم تضمين رمز مميز في الطلبات التي يتم إرسالها، ويتحقق التطبيق من وجود هذا الرمز قبل تنفيذ الطلب. هذا يجعل من الصعب على المهاجمين تنفيذ هجمات CSRF بنجاح.



**4. XSS**

طبعًا، الـ XSS (Cross-Site Scripting) هو نوع آخر من الهجمات الأمنية على تطبيقات الويب. في حالة الـ XSS، يتم استغلال الثقة المفرطة للمستخدم في المواقع التي يثق بها.

الفكرة الرئيسية للـ XSS هي حقن رموز تنفيذ البرنامج (مثل JavaScript) في صفحات الويب التي يتم عرضها للمستخدم. عندما يتم تشغيل السكريبت المشبوه، يتم تنفيذ الأوامر المضمنة فيه، ويمكن للمهاجم من خلال ذلك سرقة معلومات المستخدم، أو تغيير محتوى الصفحة، أو توجيه المستخدم إلى صفحات مزيفة للهجوم.

لمحاربة الـ XSS، يجب على المطورين اتباع ممارسات أمانية مثل تصفية وتهيئة البيانات المدخلة، وتجنب عرض البيانات المدخلة من قبل المستخدمين بدون معالجة أمنية، واستخدام العناوين الآمنة (secure headers) في رؤوس الصفحة، وتفعيل ميزات الحماية مثل Content Security Policy (CSP) التي تقيد ما يمكن تنفيذه من السكريبتات على الصفحة.

باختصار، الـ XSS يجب أن يتم مكافحته بشدة لحماية التطبيقات من الاختراقات الخبيثة وحماية خصوصية المستخدمين.A diagram of a computer

Description automatically generated

**5. difference between composer update and composer install**

composer update:

عند تشغيل composer update، Composer يقوم بتحديث جميع الحزم في ملف composer.json إلى أحدث الإصدارات المتاحة ويقوم بتحديث ملف composer.lock لتسجيل الإصدارات الجديدة.

يستخدم composer update عندما تريد تحديث جميع الحزم في مشروعك إلى أحدث الإصدارات المتاحة، بغض النظر عن الإصدارات المحددة في ملف composer.lock.

composer install:

عند تشغيل composer install، Composer يقوم بتثبيت الحزم المحددة في ملف composer.lock بالإصدارات المحددة دون تغيير. إذا لم يكن ملف composer.lock موجودًا، فسيتم استخدام ملف composer.json.

يستخدم composer install عندما تريد تثبيت الحزم بالإصدارات المحددة في ملف composer.lock، مما يجعل التنصيب أكثر استقرارًا وتنظيمًا.

بشكل عام، عند بدء مشروع جديد، يجب استخدام composer install لضمان تثبيت الحزم بالإصدارات المحددة في ملف composer.lock. وعندما تكون بحاجة لتحديث الحزم إلى أحدث الإصدارات، فيمكنك استخدام composer update.

**7. For reading (request life cycle, service provider, service container, facade)**

دورة حياة الطلب (Request Life Cycle): هذا يشير إلى سلسلة الخطوات التي يمر بها طلب HTTP في تطبيق Laravel منذ استقباله إلى توليد الاستجابة. يتضمن ذلك التوجيه (Routing)، وتنفيذ المنطق (Logic Execution)، واسترجاع العرض (View Rendering)، وإرجاع الاستجابة (Response Sending).

مزود الخدمة (Service Provider): في Laravel، مزود الخدمة هو كلاس يقوم بتهيئة خدمات التطبيق وتسجيلها في حاوية الخدمات (Service Container)، وذلك من خلال طرق مثل register() و boot().

حاوية الخدمات (Service Container): هنا، حاوية الخدمات هي القلب النابض لتطبيق Laravel. تحتوي على مفاتيح وقيم تمثل خدمات التطبيق المختلفة، ويمكن الوصول إليها بسهولة من أي مكان في التطبيق.

الواجهة (Facade): في Laravel، الواجهة هي ببساطة واجهة سهلة للوصول إلى خدمات معينة في حاوية الخدمات، وتوفر طريقة سهلة لاستخدامها دون الحاجة إلى إنشاء مثيلات من الكلاسات. تمكنك الواجهات من الوصول إلى خدمات Laravel مثل قاعدة البيانات (DB) أو البريد (Mail) بسهولة وباستخدام رمز قليل.

A diagram of a life cycle

Description automatically generated

**8. server status codes (different types of families)**

1xx - معلوماتية: هذه الأكواد توفر معلومات فقط ولا تعتبر استجابات نهائية. عادةً ما تُستخدم لتحديد أن السيرفر قد استلم الطلب ويجب أن ينتظر المزيد من المعلومات قبل الاستمرار. على سبيل المثال:

100 Continue: يُستخدم لإبلاغ العميل أن السيرفر قد استلم الجزء الأول من الطلب وينتظر المزيد.

2xx - نجاح: تشير هذه الأكواد إلى أن الطلب قد تم استلامه ومعالجته بنجاح. على سبيل المثال:

200 OK: يُستخدم لتأكيد أن الطلب قد تم معالجته بنجاح والاستجابة تحتوي على البيانات المطلوبة.

3xx - تحويل: تستخدم هذه الأكواد لتوجيه العميل إلى موارد أو مواقع جديدة. على سبيل المثال:

301 Moved Permanently: يُستخدم لإعلام العميل أن الموارد المطلوبة تم نقلها بشكل دائم إلى موقع جديد.

4xx - خطأ العميل: تشير هذه الأكواد إلى أن هناك خطأ في الطلب الذي قدمه العميل. على سبيل المثال:

400 Bad Request: يُستخدم لإشعار العميل بأن الطلب غير صحيح ولا يمكن معالجته بسبب بيانات غير صالحة.

5xx - خطأ السيرفر: هذه الأكواد تشير إلى أن هناك خطأ في السيرفر نفسه أثناء معالجة الطلب. على سبيل المثال:

500 Internal Server Error: يُستخدم لإشعار العميل بأن هناك خطأ داخلي في السيرفر يمنع معالجة الطلب بشكل صحيح.

هذه الفئات تسهل فهم حالة السيرفر والتعامل مع الأخطاء المحتملة التي يمكن أن تحدث أثناء التفاعل بين العميل والسيرفر.A colorful cubes with text

Description automatically generated

**9. if and isset directives in blade**

**1. @if Directive**

هنا الـ `@if` بيستخدم عشان تحدد إذا كنت عاوز تعرض محتوى معين ولا لا. مثلاً:

@if($condition)

<p>المحتوى ده هيتعرض لو الشرط صح.</p>

@else

<p>المحتوى ده هيتعرض لو الشرط غلط.</p>

@endif

**تقدر تستخدم `@elseif` و `@else` عشان تحط شروط إضافية:**

@if($condition1)

<p>محتوى للشرط رقم 1.</p>

@elseif($condition2)

<p>محتوى للشرط رقم 2.</p>

@else

<p>محتوى افتراضي.</p>

@endif

**2. @isset Directive:**

**الـ `@isset` بيستخدم عشان تتأكد إن المتغير معرف ومش فاضي. مثلاً:**

@isset($variable)

<p>المتغير معرف ومش فاضي.</p>

@else

<p>المتغير مش معرف أو فاضي.</p>

@endisset

**وتقدر تستخدم `@empty` للتحقق إذا كان المتغير فاضي:**

@isset($variable)

<p>المتغير معرف ومش فاضي.</p>

@empty

<p>المتغير فاضي أو مش معرف.</p>

@endisset

باستخدام هذه الحالات، تقدر تكتب قوالب بليد أكثر وضوحًا وسهولة فهم، وتتعامل مع شروط ووجود المتغيرات بشكل أكثر كفاءة.A screenshot of a computer program

Description automatically generated