

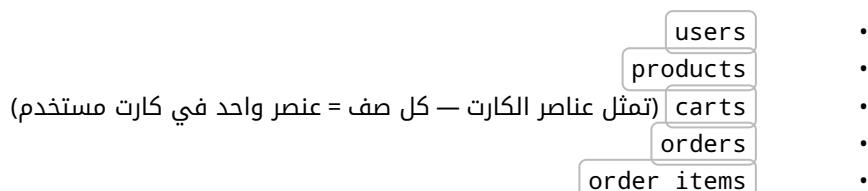
مخطط قاعدة البيانات (DB Diagram) وشرح تفصيلي

DB diagram

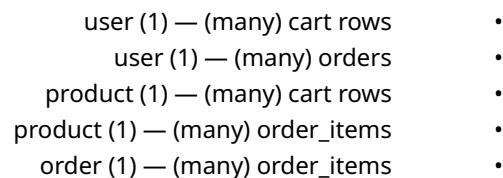
هذا المستند يشرح المخطط الموجود أعلاه (الذي أدرجته) بالتفصيل — جداول، حقول، علاقات، قواعد سلامة البيانات، واعتبارات للتنفيذ في لارافيل.

نظرة عامة سريعة

الـ schema يحتوي على الجداول الأساسية التالية:

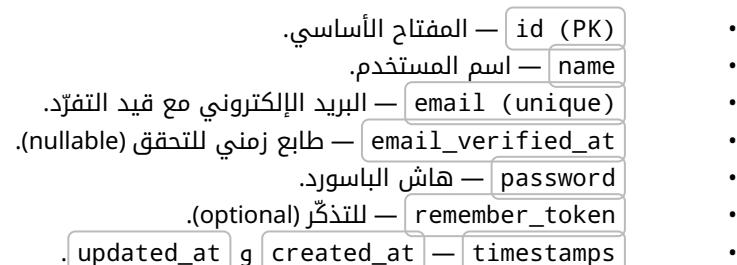


العلاقات الأساسية:



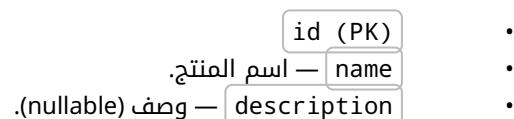
شرح الجداول (حقل بحفل)

users



تعليق: الجدول يمثل المستخدمين والمسؤول عن الربط مع الكارت والأوردرز.

products



— سعر الوحدة.	decimal(10,2) — price
— كمية المخزون الحالية.(integer)	quantity
— رمز المنتج إن وجد.	sku (unique)
boolean لتفعيل/تعطيل المنتج.	is_active
	timestamps

• هو الحقل الذي نخفضه عند إنشاء الأوردر. يجب مراعاة التزامن (transactions/locking) عند تعديله لتفادي oversell.

carts

•	id (PK)
•	user_id (FK -> users.id) — حالة الماينجرشن.
•	constrained()->onDelete('cascade') في من نفس القاعدة.
•	product_id (FK -> products.id) — عدد الوحدات في هذا العنصر.
•	quantity timestamps
•	— هذا يضمن أن كل زوج (مستخدم، منتج) يظهر مرة واحدة UNIQUE (user_id, product_id) فقط.

• مفهوم الداتا: كل صف في carts الصنوف التي تشتراك في user_id واحد.

• ملاحظة تصميمية: إن أردت فصل مفهوم "كار特" ككيان مستقل (مثلًا لخصائص الكارت ك currency id, user_id, shopping_carts يحتوي على ... user_id بدلًا من shopping_cart_id تشير إلى cart_items ثم meta...).

orders

•	id (PK)
•	order_number (unique) — رقم الطلب، يجب أن يكون فريد (مثلًا ORD-XXXX).
•	address — عنوان الشحن.
•	phone — رقم العميل.
•	total — المجموع decimal(10,2).
•	timestamps

• تعلقيات: لا يوجد حقل user_id في الماينجرشن التي عرضتها — لو طالب التطبيق، يجب عادة ربط الأوردر بالمستخدم (إضافة foreignId('user_id')->constrained()->onDelete('cascade')) حتى نعرف صاحب الطلب. إن لم يكن مقصودًا، فتأكد أن متطلباتك تسمح بأوردرات غير مرتبطة بمستخدم.

order_items

•	id (PK)
•	order_id (FK -> orders.id) onDelete cascade
•	product_id (FK -> products.id) onDelete cascade
•	quantity — عدد الوحدات في هذا الطلب.
•	price — السعر في وقت الشراء (هام لثبيت السعر لاحقًا).

timestamps

مهم لأنه يضمن أن السجل يحتفظ بسعر بيع تلك الحصة **order_items.price** تعليقات: حفظ السعر داخل حتى لو تغير سعر المنتج لاحقاً.

علاقات ER (شرح العلاقات بالتفصيل)

User -> Carts (1 - many)

يملك المستخدم عدة صنوف في جدول **carts**, كل صنف منتج واحد في الكارت. فعلياً هذه تمثل "عناصر الكارت", وليس "كاراتات متعددة".

Product -> Carts (1 - many)

كل منتج يمكن أن يكون موجوداً في كاراتات عدة مستخدمين.

Order -> OrderItems (1 - many)

كل أوردر يحتوي على عدة عناصر: **order_items** يخزن التفاصيل لكل عنصر.

Product -> OrderItems (1 - many)

المنتج نفسه يمكن أن يظهر في العديد من طلبات العملاء.

نقاط مهمة للهندسة والتنفيذ في لارايفيل

ربط أوردر بالمستخدم

أوامر إضافة **orders** إلى جدول **user_id** حتى تضمن القدرة على استرجاع أوردرات المستخدم.

```
\$table->foreignId('user_id')->constrained()->onDelete('cascade');
```

مثال في المايجريشن: **onDelete('cascade')**

التحكم بالمخزون (Stock)

عند إنشاء الأوردر: نفذ كل شيء داخل **DB::transaction()** كما فعلت.
لتتجنب حالات السباق (race conditions) عند الـ **high concurrency**:
استخدم قفل صنف (row locking) في **Eloquent**.

```
\$product = Product::where('id', \$id)->lockForUpdate()->first();
```

هذا يضمن أن عمليتي كتابة متزامنتين لن تفرضا قيمة stock خاطئة.

التخزين المؤقت (caching) وإن dexes

ضع على الحقول التي تستدل كثيراً مثل **product_id** و **user_id** **index** عادة تعمل على **foreign keys**.
الآن **index** تلقائياً.

أو quantity	فكِّر في cache لصفحة المنتج أو قائمة المنتجات لكن تأكِّد من إبطال الكاش عند تغيير .is_active	.10
	المعاملات (transactions)	.11
عند فشل أي خطوة أثناء إنشاء الأوردر، قم بعمل DB::rollBack() ومسح أي تعديل على الـ stock.	.12	
أنت بالفعل تستخدم هذا النمط — ممتاز.	.13	
	carts في unique constraint	.14
مفید يمنع تكرار صفين لنفس المنتج في نفس كارت. بدلاً من UNIQUE(user_id, product_id) عند إضافة نفس المنتج، قم بزيادة الـ quantity.	.15	
	حقل السعر في order_items	.16
تأكد من نسخ وقت الإنشاء — هذا يمنع تغيير التاريخي order_items.price إلى product.price للأسعار.	.17	
	Model relations (Eloquent)	.18
.orders (أو أضفت user_id إلىhasMany(Order::class),hasMany(Cart::class) :Users .belongsTo(Product::class),belongsTo(User::class) :Cart .belongsTo(User::class),hasMany(OrderItem::class) :Order .belongsTo(Product::class),belongsTo(Order::class) :OrderItem (إن وُجد).	.19 .20 .21 .22	

تدفق مثال عملي (Flow) — من Add to Cart إلى Create Order

. quantity	و product_id مع POST /api/auth/cart	المستخدم يضيف منتجًا:	.1
	في السيرفر: إذا وجد صف في carts، زد quantity أو عَيّره حسب الطلب.	.2	
المستخدم يعرض الكارت: carts مع منتجاتها المرتبطة.	→ تُعيد كل صفوف GET /api/auth/cart	.3	
عند إنشاء الأوردر: phone و address مع POST /api/auth/orders	السيرفر يجلب كل عناصر carts للمستخدم.	.4	
	داخل DB::transaction() لكل منتج: جلب المنتج مع lockForUpdate()	.5 .6	
إن كافي: decrement('quantity', qty) في نفس الترانزاكشن.	التحقق من stock في الـ quantity	.○ .○	
	تحضير .order_items تحديداً quantity و price مع order_items	.○	
	إنشاء السجل في orders ثم order_items	.7	
	حذف صفوف الـ carts للمستخدم	.8	
	.items_summary, total, order_number إرجاع JSON فيه:	.9	

أمثلة استعلامية (SQL / Eloquent)

جلب كارت المستخدم مع تفاصيل المنتج:

```
\$cart = Cart::with('product')->where('user_id', \$userId)->get();
```

قفل المنتج وخصم إلـ stock (داخل Transaction):

```
DB::transaction(function() use (\$productId, \$qty) {
    \$product = Product::where('id', \$productId)->lockForUpdate()->first();
    if (!\$product || \$product->quantity < \$qty) {
        throw new \Exception('Not enough stock');
    }
    \$product->decrement('quantity', \$qty);
});
```

إنشاء order_items مع order (مبسط):

```
\$order = Order::create([...]);
foreach (\$items as \$it) {
    OrderItem::create(array_merge(\$it, ['order_id' => \$order->id]));
}
```

حالات إلـ Edge / Error Handling

- المنتج مذوف أو غير موجود في وقت إنشاء الأوامر (يُرجع 404).
- عدم توافر كمية كافية (يُرجع 400 مع رسالة واضحة).
- محاولة إضافة منتج في الكارت بعد أن يصبح `is_active=false`.
- سباق على إلـ stock — حلـها بالـ `lockForUpdate()` أو باستخدام صفة `select ... for update`.
- التحقق من صلاحيات المستخدم: الوصول لـ cart يتطلب تحقق JWT.

اختبارات مقتربة (tests)

.decrement على موديل Product (Unit tests) .1

:Feature tests .2

إضافة عنصر للكارت، تحديث الكمية، حذف العنصر. .3

إنشاء الأوامر: نجاح المسار (decrments, clears cart, returns order_number) .4

محاولة إنشاء أوامر عندما المخزون غير كاف (assert 400) .5

ملاحظات نهائية

- إن أردت أن أعدل المايجريشنز لربط `shopping_carts` منفصل، أقدر أكتب لك لا `.models` و لا `migration`.
- لو تدب، أحول هذا المخطط إلى **Mermaid diagram** لوضعه مباشرة داخل `README.md` كنص (سهل القراءة ويدعم GitHub). أيضاً أقدر أعمل صورة عالية الجودة لا `diagram`.

هل تريد التحويل إلى **Mermaid diagram** لنسخه في `README`؟ أو أعمل التغييرات التالية مباشرة في الكود:
(ضع علامة ✓ على الخيارات): - [✓] أعمل جدول `shopping_carts` إلى `orders` [✓] أضيف `user_id` منفصل
- [✓] أضف `lockForUpdate()` في الكود الحالي [✓] `cart_items` و