API Tesztelési Stratégia Megvalósítása - Webshop Backend
Feladatleírás
A feladat célja egy API tesztelési stratégia kidolgozása és megvalósítása egy webshop backend alkalmazásra. A webshop backend funkciói közé tartozik a termékek kezelése, a felhasználói fiókok kezelése, a rendeléskezelés és a kosár funkció. A tesztelési stratégia célja a backend API-ok megbízhatóságának, teljesítményének és alapszintű biztonságának biztosítása.

API Tesztelési Stratégia

1. Bevezetés

Ez a dokumentum a webshop backend API tesztelési stratégiáját mutatja be. Célja, hogy meghatározza a tesztelési célokat, módszereket, eszközöket és felelősségi köröket annak érdekében, hogy biztosítsuk az API megbízhatóságát, funkcionalitását, biztonságát és teljesítményét.

🔹 **Cél**: A tesztelési stratégia célja a webshop backend API minőségének biztosítása a fejlesztési és bevezetési folyamat során.

🔹 **Hatókör**: A stratégia lefedi az összes API végpont funkcionális és nem-funkcionális tesztelését, ideértve a termékkezelést, felhasználókezelést, kosár és rendelés funkciókat.

2. Tesztelési célok

- 🔹 A funkcionális követelmények teljes körű ellenőrzése (pl. CRUD műveletek, jogosultságok).
- 🔹 Teljesítmény és válaszidők mérése, különböző terhelés alatt.
- 🔹 Biztonsági rések és sebezhetőségek azonosítása (pl. jogosulatlan hozzáférés, input validáció).
- 🔹 A felhasználói élmény javítása.
- 🔹 Hibakezelés és visszajelzések konzisztenciájának ellenőrzése.
- 🔹 A rendszer megbízhatóságának, stabilitásának és skálázhatóságának biztosítása.

2.1. Funkcionális tesztelés

🔹 **Cél**: Az API végpontok alapvető működésének és teljes funkcionalitásának ellenőrzése valós forgatókönyvek alapján.
✅ **Megvalósítás**:
- CRUD műveletek: Termékek, rendelések és felhasználók kezelése.
- Sikeres és sikertelen kérések: 200, 400, 401, 403, 500 válaszok kezelése.
- Állapotváltozások: Kosárból rendelés, rendelés státusz módosítása.
- Adatmodell validáció (pl. kötelező mezők megléte).
- Üzleti logika helyessége (pl. kedvezmények számítása).
- Hibakezelés (pl. hiányzó vagy hibás adatok kezelése).

⚙️ **Javasolt eszközök:** Postman, Swagger

2.2. Integrációs tesztelés

🔹 **Cél**: API végpontok és komponensek közötti adatáramlás tesztelése.
✅ **Megvalósítás**:
- Felhasználói folyamatok: Bejelentkezés → Kosár → Rendelés → Fizetés.
- Adatbázis műveletek: Adatok helyes mentése és visszaolvasása.

⚙️ **Javasolt eszközök:** Postman, GitHub Actions / GitLab CI / Jenkins a pipeline részeként

2.3. Terhelés- és teljesítmény tesztelés

🔹 **Cél**: A backend teljesítményének és skálázhatóságának vizsgálata.
✅ **Megvalósítás**:
- API válaszidők normál és nagy terhelés alatt.
- 100, 500, 1000 párhuzamos felhasználó kezelése.
- Bottleneck-ek és skálázhatósági problémák feltárása.

⚙️ **Javasolt eszközök:** JMeter

2.4. Biztonsági tesztelés

🔹 **Cél**: Alapvető biztonsági sérülékenységek azonosítása.
✅ **Megvalósítás**:
- Autentikáció és jogosultságok: Felhasználói tokenek érvényessége, szerepkörök kezelése.
- Adatbiztonság: Adatbázisban tárolt jelszavak titkosítása.
- Támadások elleni védelem: SQL injection, XSS, brute-force támadások.

⚙️ **Javasolt eszközök:** Postman (esetleg SOAPUI), OWASP ZAP, Burp Suite

2.5. CI/CD és Automatizálás

🔹 **Cél**: Gyors visszacsatolás, tesztek automatikus futtatása minden fejlesztés után.
⚙️ **Javasolt eszközök:** JMeter

3. Tesztelési forgatókönyvek

🔹 **Funkcionális tesztelés:**
✅ Pozitív és negatív tesztelési forgatókönyvek kidolgozása.
✅ Tesztelési esetek a felhasználói interakciók ellenőrzésére.

🔹 **Nem-funkcionális tesztelés:**
✅ Teljesítmény tesztelési forgatókönyvek (pl válaszidők mérése különböző végpontokon).
✅ Biztonsági tesztelési forgatókönyvek (pl jogosultság-ellenőrzés).
✅ Megbízhatósági és stabilitási tesztelési forgatókönyvek (pl folyamatos hívások és éles szimulációk hosszabb időn keresztül).

4. Tesztelési eszközök

🔹 **Tesztelési eszközök kiválasztása:** A megfelelő tesztelési eszközök és keretrendszerek kiválasztása a projekt igényei alapján. Figyelembe kell venni az előzetes technikai ismereteket.
🔹 **Integráció:** A tesztelési eszközök integrálása a CI/CD folyamatokba.

⚙️ **Javasolt eszközök:**
- Automatizált tesztek: Postman, REST Assured
- Teljesítményteszt: JMeter
- Biztonsági teszt: OWASP ZAP, Burp Suite
- Integráció: GitHub Actions / GitLab CI / Jenkins a pipeline részeként

5. Tesztelési folyamat

🔹 **Tesztelési forgatókönyvek kidolgozása**: A tesztelési forgatókönyvek részletes kidolgozása.
⚙️ **Javasolt eszköz:**
Test design: Confluence
Részletes tesztesetek: JIRA/Zephyr

🔹 **Tesztelés végrehajtása**: A kiválasztott eszközök segítségével végrehajtani a teszteket.
🔹 **Eredmények rögzítése:** A tesztelési eredmények és a talált hibák rögzítése.
⚙️ **Javasolt eszköz:**
Tesztek dokumentációja és futtatási eredmények: JIRA, Zephyr

🔹 **Jelentés készítése**: A tesztelési folyamat összegzése és a javasolt javítások.
✅ **Hiba nyomon követése:**

- JIRA használata a hibák kezelésére.
- Kritikus hibák azonnali javítása.
- Prioritások meghatározása:
  🔴 Kritikus: API nem válaszol, adatvesztés.
  🟠 Magas: Jogosultsági hiba, rendeléskezelési probléma.
  🟡 Közepes: UI/UX problémák, kisebb validációs hibák.
  🔵 Alacsony: Ritka edge case hibák.

✅ **Jelentések és dashboardok:**

- API teljesítmény és teszt lefedettség grafikonok (Grafana)
- Automatikus tesztek eredményei CI/CD pipeline-ban.

6. Tesztelési Ütemterv

🔹 **Időkeret**: A tesztelési folyamat becsült időtartama.

- Sprintenkénti automatizált regressziós tesztfuttatás
- Kézi tesztelés a funkciók fejlesztése után
- Becslés: Story point, T-shirt sizing

🔹 **Feladatok**: A feladatok ütemezése és felelősségek meghatározása.

- Sprint eleje: Statikus tesztelés, a tervezés korai szakaszától kezdve (Refinement), teszttervezés, tesztkörnyezet felállítása, tesztadatok előállítása
- Sprint közepe: A tesztelési forgatókönyvek futtatása a fejlesztés befejezésével kezdődően
- Sprint végén: Terheléses- és biztonsági tesztek, regressziós tesztek, automata tesztek

🔹 **Felelősségek**:

- QA: Tesztelési terv és végrehajtás
- Dev: Bugfix, unit tesztek írása
- DevOps: CI/CD beállítás és karbantartás

7. Kockázatok és kihívások

🔹 **Kockázatok azonosítása:** A tesztelési folyamat során felmerülő kockázatok és kihívások azonosítása.

- Hiányos specifikáció
- Tesztadat-kezelés problémái
- Tesztek és fejlesztések párhuzamosítása nehézségeket okozhat

🔹 **Kockázatkezelés**: A kockázatok minimalizálására irányuló intézkedések kidolgozása.
- Korai bevonás a tervezésbe ("shift left")
- Mock adatbázis és izolált környezet használata
- CI bevezetése már a fejlesztés korai szakaszában

8. Folyamatos fejlesztés

🔹 **Visszajelzés gyűjtése:** A tesztelési eredmények és tapasztalatok visszajelzése a fejlesztési csapat számára, QA- és fejlesztői visszajelzések alapján a tesztesetek finomítása

🔹 **Folyamatos fejlesztés:** A tesztelési folyamatok folyamatos felülvizsgálata és javítása
- Újabb hibák automatikus lefedése regressziós teszttel
- Tesztkód refaktorálása
- Tesztelési eszközök és folyamatok iteratív fejlesztése

Funkció		Javasolt tesztesetek		Manuális	Automatizálás?
1. Termékek Kezelése					

Termékek listázása	o GET /products
	Példa válasz:
	[
	{
	"id": 1,
	"title": "Product 1",
	"description": "Description of Product 1",
	"price": 29.99,
	"stock": 100,
	"category": "Electronics"
	}
	]				

		tests for data	(+)	Successful GET {endpoint} with **valid data** - Check whether the response contains all data according to Swagger. NULL values may not come back.	**yes**
		tests for pagination	(+)	Successful GET {endpoint} with **valid pagination** (?page=1&limit=10)	**yes**
			(-)	Unsuccessful GET {endpoint} with **invalid** pagination (?page=-1)	
		tests for filtering	(+)	Successful GET {endpoint} with **valid filter** (?category=electronics)	**yes**
			(-)	Unsuccessful GET {endpoint} with **invalid** filter value (?category=ufonautics) **HTTP 200 OK - empty list**	
		tests for ordering	(+)	Successful GET {endpoint} with **valid ordering** (?sort=price_asc)	**yes**
		load tests	(+)	Successful GET {endpoint} with **valid data - over 10000 products**	**yes from time to time**
			(+)	Successful GET {endpoint} with **valid data - parallel requests** (min 100)	**yes from time to time**
		security tests	(+)	Unsuccessful **SQL injection** (?search=' OR 1=1 --)	**yes from time to time**

Meglévő termék frissítése	o PUT /products/{id}
	Példa kérés:
	{
	"id": 1,
	"title": "Updated Product",
	"description": "Updated description",
	"price": 24.99,
	"stock": 150,
	"category": "Electronics"
	}
	Példa válasz:
	{
	"id": 1,
	"title": "Updated Product",
	"description": "Updated description",
	"price": 24.99,
	"stock": 150,
	"category": "Electronics"
	}				

		tests for data	(+)	Successful PUT {endpoint} with **all data** - Check whether the response contains every data according to Swagger. NULL values may not come back. - Check the updated created record in DB (if possible)	**yes**

			(+)	Successful PUT {endpoint} with the **same previous data** - Check the updated created record in DB (if possible)	
			(+)	Successful PUT {endpoint} **without the not required data**	
			(+)	Successful PUT {endpoint} with the **not required data one by one**	
			(-)	Unsuccessful PUT {endpoint} with **not existing ID** in path param **HTTP 404**	**yes**
			(-)	Unsuccessful PUT {endpoint} with **ID mismatch** - not related ID in path param and in JSON body	**yes**
			(-)	Unsuccessful PUT {endpoint} with **empty/missing ID** in path param	
			(-)	Unsuccessful PUT {endpoint} with **empty/missing** required data one by one	**yes**
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful PUT {endpoint} with **wrong format NUMBER** (contains SPACE, just SPACE, with commas, special characters, letters, more than X digits, < or > than min-max)	**yes**
			(-)	Unsuccessful PUT {endpoint} with **wrong format STRING**	**yes**
		tests for **ADMIN** user authentication	(-)	Unsuccessful PUT {endpoint} with **empty/missing ADMIN user** login parameters (empty accessToken, missing accessToken) **HTTP 401**	**yes**
			(-)	Unsuccessful PUT {endpoint} with **invalid ADMIN user** login parameters (invalid accessToken: regular user's sessionToken, expired) **HTTP 403**	**yes**
		security tests	(+)	Unsuccessful **SQL injection** (?search=' OR 1=1 --)	**yes from time to time**

Termék törlése	o DELETE /products/{id} Válasz: Státuszkód: 204 No Content				
		tests for data	(+)	Successful DELETE {endpoint} with **valid data (product ID in path param)** **HTTP 204** - Check whether the valid record has been deleted from DB.	**yes**
		tests for invalid data	(-)	Unsuccessful DELETE {endpoint} data with **already deleted ID** (product ID in path param) **HTTP 404**	**yes**
			(-)	Unsuccessful DELETE {endpoint} data with **not existing ID** (product ID in path param)	**yes**
			(-)	Unsuccessful DELETE {endpoint} data with **wrong format ID** (product ID in path param)	
		tests for empty /missing data	(-)	Unsuccessful DELETE {endpoint} data with **empty/missing ID** (product ID in path param)	
		tests for **ADMIN** user authentication	(-)	Unsuccessful DELETE {endpoint} with **empty/missing ADMIN user** login parameters (empty accessToken, missing accessToken) **HTTP 401**	**yes**
			(-)	Unsuccessful DELETE {endpoint} with **invalid ADMIN user** login parameters (invalid accessToken: regular user's sessionToken, expired) **HTTP 403**	**yes**
		security tests	(+)	Unsuccessful **SQL injection** (DELETE /products/1;DROP TABLE products) - Check that DB hasn't been affected.	**yes from time to time**

Funkció		Javasolt tesztesetek		Manuális	Automatizálás?
2. Felhasználói Fiókok Kezelése					

Felhasználói regisztráció	o POST /users/register
	Példa kérés:
	{
	"username": "newuser",
	"password": "securepassword",
	"email": "user@example.com"
	}
	Példa válasz:
	{
	"id": 1,
	"username": "newuser",
	"email": "user@example.com"
	}				

		tests for data	(+)	Successful POST {endpoint} with **all data** - Check whether the response contains every data according to Swagger. **Password must not come back.** - Check the newly created record in DB (if possible) - Check whether a notification has been sent.	**yes**
			(-)	Unsuccessful **repeated** POST {endpoint} with the **same previous data**	
			(-)	Unsuccessful POST {endpoint} with **empty/missing** required data one by one	**yes**
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful POST {endpoint} **with wrong format STRING**	**yes**
			(-)	Unsuccessful POST {endpoint} with **empty body**	**yes**
		security tests	(+)	Unsuccessful **SQL injection** (?search=' OR 1=1 --)	**yes from time to time**

Bejelentkezés	o POST /users/login
	Példa kérés:
	{
	"username": "newuser",
	"password": "securepassword"
	}
	Példa válasz:
	{
	"token": "jwt-token-here"
	}				

		tests for data	(+)	Successful POST {endpoint} with **all data** - Check whether the response contains every data according to Swagger. **Password must not come back.** - Check the newly created record in DB (if possible) - Check whether a notification has been sent.	**yes**
			(-)	Unsuccessful POST {endpoint} with **empty/missing** required data one by one	**yes**
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful POST {endpoint} **with wrong format STRING**	**yes**

			(-)	Unsuccessful POST {endpoint} with **empty body**	**yes**
			(-)	Unsuccessful POST {endpoint} with **extra key-value**	
		tests for user authentication	(-)	Unsuccessful POST {endpoint} with **empty/missing** login parameters (empty username/password, missing username/password) **HTTP 401**	**yes**
			(-)	Unsuccessful POST {endpoint} with **not related** username/password **HTTP 401**	**yes**
		security tests		brute-force hack Is there a limit or too many requests?	
			(+)	Unsuccessful **SQL injection** (?search=' OR 1=1 --)	**yes from time to time**

Felhasználói profil frissítése	o PUT /users/{id}
	Példa kérés:
	{
	"email": "updateduser@example.com",
	"password": "newsecurepassword"
	}
	Példa válasz:
	{
	"id": 1,
	"username": "newuser",
	"email": "updateduser@example.com"
	}				

		tests for data	(+)	Successful PUT {endpoint} with **all data** - Check whether the response contains every data according to Swagger. **Password must not come back.** - Check the updated record in DB (if possible)	**yes**
			(+)	Successful PUT {endpoint} with the **same previous data** - Check the updated created record in DB (if possible)	
			(+)	Successful PUT {endpoint} with the **not required data one by one**	
			(-)	Unsuccessful PUT {endpoint} with **not existing ID** in path param **HTTP 404**	**yes**
			(-)	Unsuccessful PUT {endpoint} with **empty/missing ID** in path param	
			(-)	Unsuccessful PUT {endpoint} with **empty/missing** required data one by one	**yes**
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful PUT {endpoint} with **wrong format STRING**	**yes**
		tests for user authentication	(-)	Unsuccessful POST {endpoint} with **empty/missing** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful DELETE {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
		security tests	(+)	Unsuccessful **SQL injection** (?search=' OR 1=1 --)	**yes from time to time**
			(-)	Unsuccessful PUT {endpoint} with **ID mismatch** - not related data in path param and in header data	**yes**

Jelszó visszaállítása	o POST /users/reset-password
	Példa kérés:
	{
	"email": "user@example.com"
	}
	Példa válasz:
	{
	"message": "Password reset link sent to email."
	}				

		tests for data	(+)	Successful POST {endpoint} with **all data** - Check whether the response contains every data according to Swagger. NULL values may not come back. - Check whether a notification has been sent. - Check whether the sent link is functioning.	**yes**
			(-)	Unsuccessful POST {endpoint} with **not existing email**	**yes**
			(-)	Unsuccessful POST {endpoint} with **empty/missing** required data one by one	**yes**
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful POST {endpoint} **with wrong format STRING**	**yes**
			(-)	Unsuccessful POST {endpoint} with **empty body**	
			(-)	Unsuccessful POST {endpoint} with **extra key-value**	
		tests for user authentication	(-)	Unsuccessful POST {endpoint} with **empty/missing** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful DELETE {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
		security tests	(+)	Unsuccessful **SQL injection** (?search=' OR 1=1 --)	**yes from time to time**
				brute-force hack Is there a limit or too many requests?	**yes from time to time**

Funkció		Javasolt tesztesetek		Manuális	Automatizálás?
3. Kosár Funkció					

Kosár tartalmának megtekintése	o GET /cart
	Példa válasz:
	{
	"items": [
	{
	"productId": 1,
	"quantity": 2
	}
	],
	"total": 59.98
	}				

		tests for data	(+)	Successful GET {endpoint} with **valid data - one item in shopping cart** - Check whether the response contains all data according to Swagger. NULL values may not come back. - Check whether the **price calculation is correct**: The product of each item's price and quantity is summed into the total field.	**yes**
			(+)	Successful GET {endpoint} with **valid data - multiple items in shopping cart** - Check whether the response contains all data according to Swagger. NULL values may not come back. - Check whether the **price calculation is correct**: The product of each item's price and quantity is summed into the total field.	**yes**
			(+)	Successful GET {endpoint} with **valid data - empty shopping cart** - Check whether the response contains all data according to Swagger. NULL values may not come back..	
		tests for user authentication **(applies only for registered user's cart)**	(-)	Unsuccessful GET {endpoint} with **empty/missing** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful GET {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
			(-)	Unsuccessful GET {endpoint} with **not related** user/cart **HTTP 401**	**yes**
		load tests	(+)	Successful GET {endpoint} with **valid data - over 10000 products**	**yes from time to time**
			(+)	Successful GET {endpoint} with **valid data - parallel requests** (min 100)	**yes from time to time**

Termék hozzáadása a kosárhoz	o POST /cart/add
	Példa kérés:
	{
	"productId": 1,
	"quantity": 2
	}
	Példa válasz:
	{
	"message": "Product added to cart."
	}				

		tests for data	(+)	Successful POST {endpoint} with **all data - one product** - Check whether the response contains every data according to Swagger. NULL values may not come back. - Check the newly created record in DB (if possible)	**yes**
			(+)	Successful POST {endpoint} with **all data - multiple product** - Check whether the response contains every data according to Swagger. NULL values may not come back. - Check the newly created record in DB (if possible)	
			(+)	Successful POST {endpoint} **without the not required data**	
			(+)	Successful POST {endpoint} with the **not required data one by one**	
			(+)	Successful **repeated** POST {endpoint} with the **same previous data**	
			(-)	Unsuccessful POST {endpoint} with **empty/missing** required data one by one	**yes**
			(-)	Unsuccessful POST {endpoint} with **not existing productId**	
			(-)	Unsuccessful POST {endpoint} with **insufficient quantity** (more than in stock)	
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful POST {endpoint} with **wrong format NUMBER** (contains SPACE, just SPACE, with commas, special characters, letters, more than X digits, < or > than min-max)	**yes**
			(-)	Unsuccessful POST {endpoint} **with wrong format STRING**	**yes**
			(-)	Unsuccessful POST {endpoint} with **empty body**	
			(-)	Unsuccessful POST {endpoint} with **extra key-value**	
		tests for user authentication **(applies only for registered user's cart)**	(-)	Unsuccessful POST {endpoint} with **empty/missing user** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful POST {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
		load tests	(+)	Successful GET {endpoint} with **valid data - over 10000 products**	**yes from time to time**
			(+)	Successful GET {endpoint} with **valid data - parallel requests** (min 100)	**yes from time to time**

Termék eltávolítása a kosárból	o DELETE /cart/remove
	Példa kérés:
	{
	"productId": 1
	}
	Példa válasz:
	{
	"message": "Product removed from cart."
	}				

		tests for data	(+)	Successful DELETE {endpoint} with **valid data - one product** - Check whether the response contains every data according to Swagger. - Check whether the valid record has been deleted from DB.	**yes**
			(+)	Successful DELETE {endpoint} with **valid data - one product from multiple different products** - Check whether the response contains every data according to Swagger. - Check whether the valid record has been deleted from the shopping cart.	
			(+)	Successful DELETE {endpoint} with **valid data - one product from multiple same products** - Check whether the response contains every data according to Swagger. - Check whether the correct quantity remained in the shopping cart.	
		tests for invalid data	(-)	Unsuccessful DELETE {endpoint} data with **already deleted ID** **HTTP 404**	**yes**
			(-)	Unsuccessful DELETE {endpoint} data with **not existing ID**	**yes**
			(-)	Unsuccessful DELETE {endpoint} data with **wrong format productId**	
		tests for empty /missing data	(-)	Unsuccessful DELETE {endpoint} data with **empty/missing productId**	
		tests for user authentication **(applies only for registered user's cart)**	(-)	Unsuccessful POST {endpoint} with **empty/missing user** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful POST {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
		reliability tests	(+)	Successful querying the cart after deletion - Use the GET /cart endpoint to verify that the product has been successfully removed.	
		load tests	(+)	Successful GET {endpoint} with **valid data - multiple requests in quick succession** (system stability and response time)	**yes from time to time**

Funkció		Javasolt		Manuális	Automatizálás?

		tesztesetek:			
4. Rendeléskezelés					

Rendelés létrehozása	o POST /orders
	Példa kérés:
	{
	"cartItems": [
	{
	"productId": 1,
	"quantity": 2
	}
	],
	"shippingAddress": "123 Main St, Anytown, USA"
	}
	Példa válasz:
	{
	"orderId": 1,
	"status": "Processing"
	}				

		tests for data	(+)	Successful POST {endpoint} with **all data - one product in cart** - Check whether the response contains every data according to Swagger. NULL values may not come back. - Check the newly created record in DB (if possible) - Check the updated inventory record in DB (if possible)	**yes**
			(+)	Successful POST {endpoint} with **all data - multiple products in cart** - Check whether the response contains every data according to Swagger. NULL values may not come back.	
			(+)	Successful POST {endpoint} **without the not required data**	
			(+)	Successful POST {endpoint} with the **not required data one by one**	
			(+)	Successful **repeated** POST {endpoint} with the **same previous data**	
			(-)	Unsuccessful POST {endpoint} with **empty/missing** required data one by one	**yes**
			(-)	Unsuccessful POST {endpoint} with **not existing productId**	
			(-)	Unsuccessful POST {endpoint} with **insufficient quantity** (more than in stock)	
		tests for wrong format **precondition: exact regex for each value**	(-)	Unsuccessful POST {endpoint} with **wrong format NUMBER** (contains SPACE, just SPACE, with commas, special characters, letters, more than X digits, < or > than min-max)	**yes**
			(-)	Unsuccessful POST {endpoint} **with wrong format STRING**	**yes**
			(-)	Unsuccessful POST {endpoint} with **empty body**	
			(-)	Unsuccessful POST {endpoint} with **extra key-value**	
		tests for user authentication **(applies only for registered user's cart)**	(-)	Unsuccessful POST {endpoint} with **empty/missing user** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**

			(-)	Unsuccessful POST {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
		load tests	(+)	Successful POST {endpoint} with **valid data - over 10000 products**	**yes from time to time**
			(+)	Successful POSTT {endpoint} with **valid data - parallel requests** (min 100)	**yes from time to time**
		reliability tests	(+)	Successful querying the cart after confirmation - Use the GET /orders/{id} endpoint to verify that the order details match with what was submitted.	

Rendelés állapotának lekérdezése	o GET /orders/{id}
	Példa kérés:
	Státuszkód: 200 OK
	Példa válasz:
	{
	"orderId": 1,
	"status": "Processing",
	"items": [
	{
	"productId": 1,
	"quantity": 2
	}
	],
	"total": 59.98
	}				

		tests for data	(+)	Successful GET {endpoint} with **valid ID** in path param **- one item in order** - Check whether the response contains all data according to Swagger. NULL values may not come back. - Check whether the **price calculation is correct**: The product of each item's price and quantity is summed into the total field.	**yes**
			(+)	Successful GET {endpoint} with **valid ID** in path param **- multiple items in order** - Check whether the response contains all data according to Swagger. NULL values may not come back. - Check whether the **price calculation is correct**: The product of each item's price and quantity is summed into the total field.	**yes**
			(-)	Unsuccessful GET {endpoint} with **not existing ID** in path param	**yes**
			(-)	Unsuccessful GET {endpoint} with **previously deleted ID** in path param	**yes**
			(-)	Unsuccessful GET {endpoint} with **empty/missing ID** in path param	
		tests for wrong format **precondition: exact regex for each value**		Unsuccessful GET {endpoint} with **wrong format ID** in path param	
		tests for user authentication **(applies only for registered user's cart)**	(-)	Unsuccessful GET {endpoint} with **empty/missing** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful GET {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
			(-)	Unsuccessful GET {endpoint} with **not related** user/orderId **HTTP 401**	**yes**
		load tests	(+)	Successful GET {endpoint} with **valid data - over 10000 products in order**	**yes from time to time**

		reliability tests	(+)	Successful GET {endpoint} with **valid data - parallel requests** (min 100) - To ensure that the same data is returned each time (if no order updates happen in-between)	

Rendelés törlése	o DELETE /orders/{id} Kérés: Státuszkód: 204 No Content				
		tests for data	(+)	Successful DELETE {endpoint} with **valid data (order ID in path param)** **HTTP 204** - Check whether the valid record has been deleted from DB.	**yes**
		tests for invalid data	(-)	Unsuccessful DELETE {endpoint} data with **already deleted ID** (order ID in path param) **HTTP 404**	**yes**
			(-)	Unsuccessful DELETE {endpoint} data with **not existing ID** (order ID in path param)	**yes**
			(-)	Unsuccessful DELETE {endpoint} data with **wrong format ID** (order ID in path param)	
		tests for empty /missing data	(-)	Unsuccessful DELETE {endpoint} data with **empty/missing ID** (order ID in path param)	
		tests for user authentication **(applies only for registered user's cart)**	(-)	Unsuccessful GET {endpoint} with **empty/missing** login parameters (empty sessionToken, missing sessionToken) **HTTP 401**	**yes**
			(-)	Unsuccessful GET {endpoint} with **invalid user** login parameters (invalid sessionToken, expired) **HTTP 403**	**yes**
			(-)	Unsuccessful GET {endpoint} with **not related** user/orderId **HTTP 401**	**yes**
		security tests	(+)	Unsuccessful **SQL injection** (DELETE /orders/1;DROP TABLE products) - Check that DB hasn't been affected.	**yes from time to time**