Laravel Tesztelés



Tartalom

[1. Laravel keretrendszer 2](#_Toc132821568)

[2. Laravel tesztek 2](#_Toc132821569)

[3. Smart24 3](#_Toc132821570)

[4. Laragon 4](#_Toc132821571)

[5. Unit tests 4](#_Toc132821572)

[5.1 Modell tests 5](#_Toc132821573)

[5.2 Controller tests 5](#_Toc132821574)

[6 Feature tests 8](#_Toc132821575)

[7 Teszt eredmények 9](#_Toc132821576)

[8 Irodalomjegyzék 10](#_Toc132821577)

Készítette:

**Lukács Szabolcs, Aut. IV.**

# Laravel keretrendszer

A Laravel egy nyílt forráskódú, ingyenes, modern webalkalmazás-keretrendszer, melyet Taylor Otwell fejlesztett ki. A PHP nyelven íródott keretrendszer sok alapvető funkciót és szolgáltatást tartalmaz, amelyek lehetővé teszik a gyors és hatékony webalkalmazások fejlesztését. A Laravel magas szintű biztonságot és támogatást kínál, így könnyű és hatékony fejlesztést tesz lehetővé. A keretrendszer nagyon népszerű, és használható kis- és nagy projektekhez egyaránt.

# Laravel tesztek

A Laravel keretrendszer nagyon erős tesztelési támogatást nyújt, amely lehetővé teszi a webalkalmazások hatékony tesztelését és hibák kiszűrését. A Laravel-ben használható tesztek PHPUnit-ra épülnek, ami egy PHP alapú tesztelő keretrendszer.

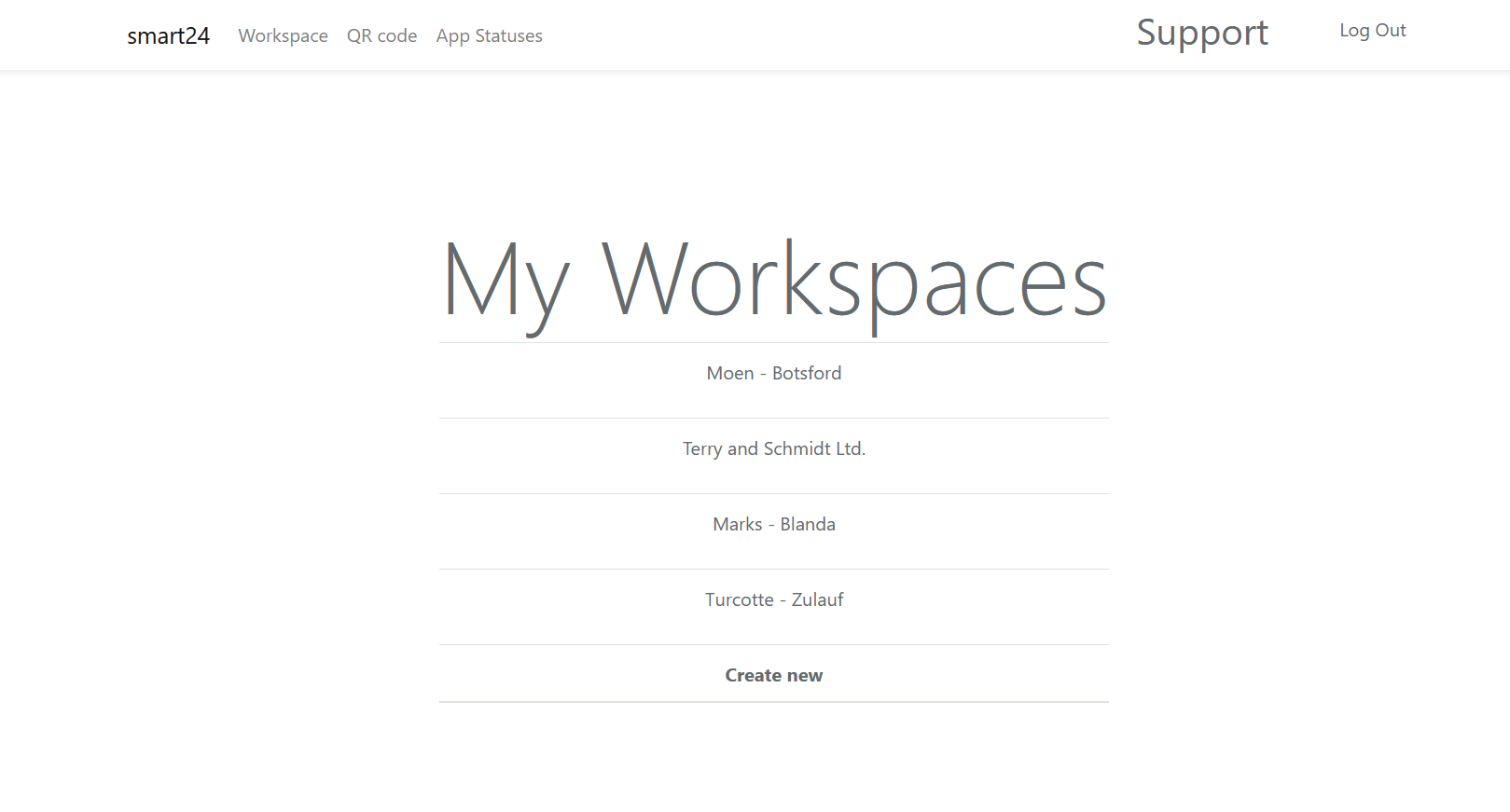
A Laravel tesztelési funkciója lehetővé teszi a különböző tesztesetek definiálását, amelyek a webalkalmazások egyes részeinek tesztelésére szolgálnak. Például a tesztelhető részek lehetnek az adatbázis műveletek, az adatok validálása, a felhasználói felület működése, a HTTP kérések és válaszok tesztelése, stb.

A Laravel tesztelési funkciója segítségével a fejlesztők automatikusan tudnak tesztelni, így időt és pénzt spórolva meg, valamint biztonságot és megbízhatóságot biztosítanak a webalkalmazások számára. A Laravel tesztelési funkciója számos előre definiált tesztosztályt és tesztelési lehetőséget biztosít, amelyek lehetővé teszik a fejlesztők számára, hogy hatékonyan és könnyen tesztelhessék a kódjukat.

A tesztek futtatása egyszerű a Laravel-ben. A tesztelési funkcióhoz először létre kell hozni egy tesztesetet, majd definiálni a teszteset műveleteit, majd futtatni a teszteket a PHPUnit segítségével.

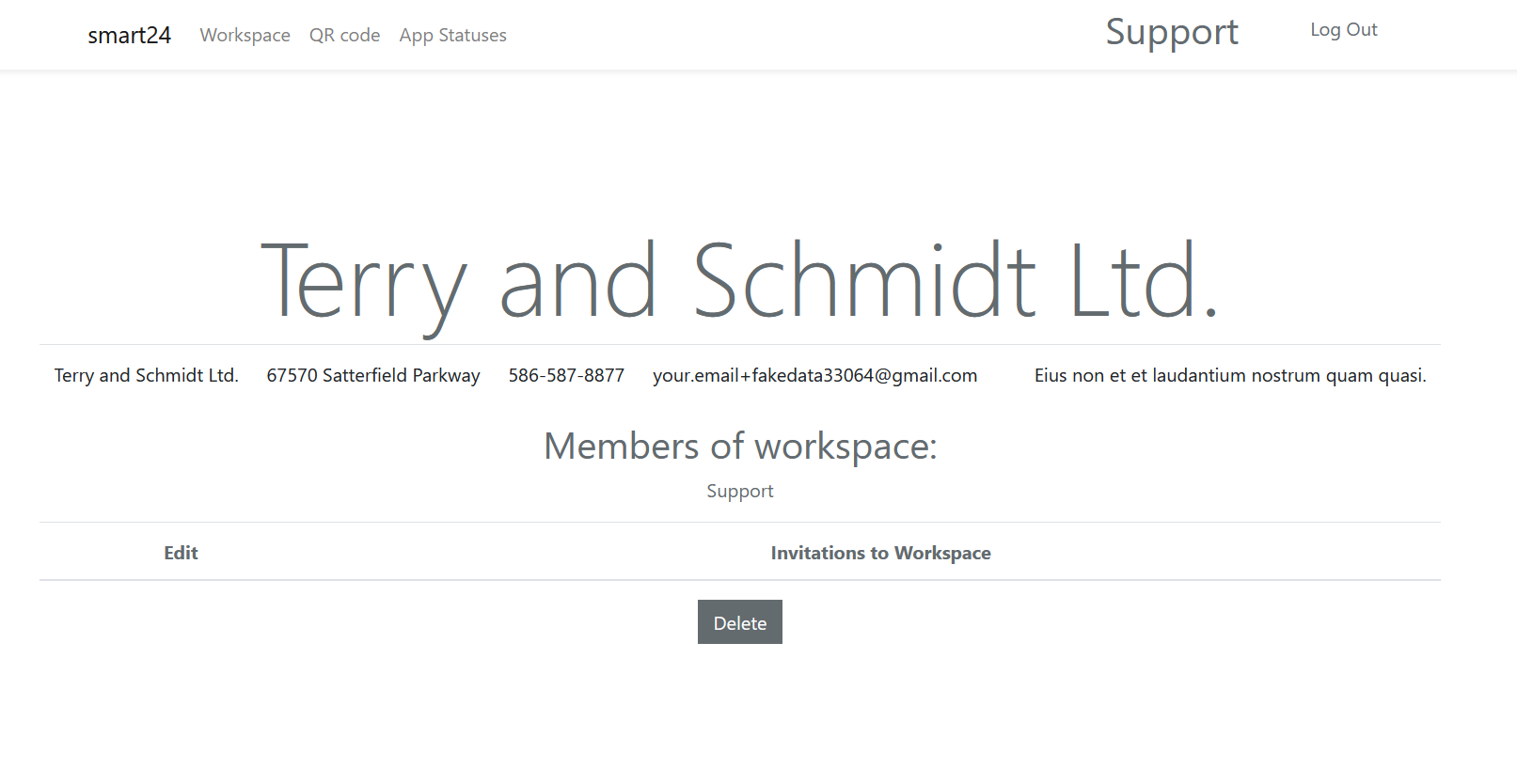
# Smart24

A Smart24 egy webalkalmazás, amely egy kisvállalkozások számára kínál hasznos funkciókat. A projekt egy szakmai gyakorlaton kezdődött el, ahol egy webfejlesztő cég keretein belül folyt a fejlesztés. Az oldal célja az, hogy egyszerű és hatékony felületet biztosítson a felhasználók számára, akik sok hasznos funkciót használhatnak.



ábra 3.1. Példaként létrehozott munkaterek

Az oldal jelenlegi funkciói közé tartozik a felhasználói regisztráció és bejelentkezés, amely lehetővé teszi a felhasználók számára, hogy hozzáférjenek a szolgáltatásokhoz. A felhasználók munkaterületeket hozhatnak létre, szerkeszthetik és törölhetik azokat, valamint alkalmazásállapotokat hozhatnak létre, szerkeszthetik és törölhetik azokat. Úgynevezett appstatusokat hozhatnak létre, ahol bizonyos funkciók állapotát figyelhetik, láthassák a módosítások dátumát. Az oldal további funkciója a QR-kód generálása, amely lehetővé teszi a felhasználók számára, hogy egyszerűen és gyorsan hozzáférjenek a weboldalukhoz.



ábra 3.2. Egy munkatérhez tartozó adatok

Az oldal számos hasznos funkciót biztosít a felhasználók számára, amelyek könnyen kezelhetők és megfelelnek a kisvállalkozások igényeinek. Az oldal funkciói könnyen elérhetők a felhasználók számára, akik gyorsan és hatékonyan tudnak dolgozni az oldalon. A Smart24 egy nagyszerű lehetőség kisvállalkozások számára, akik hatékonyan szeretnék kezelni a munkafolyamataikat.

# Laragon

A weboldalt a Laragon alkalmazás segítségével teszteltem a saját számítógépemen. A Laragon egy könnyen használható fejlesztőkörnyezet, amely lehetővé teszi, hogy könnyedén futtassunk egy lokális szervert a saját gépünkön.

# Unit tests

A unit tesztjeim két fő részre vannak osztva: Modells, Controllers. A Modell-ben a modellek közti kapcsolatokat ellenőrzöm, például, hogy egy felhasználóhoz több munkatér is tartozhat. A Controllers-ben leginkább az oldal funkciói tesztelem. Például a QR generálás funkciót.

## Modell tests

4 fő osztályom van:

* AppStatusTest
* UserTest
* WorkSpaceInvitationTest
* WorkSpaceTest

Ezekben rendszerint létrehozok egy-egy példányt az adatbázisban, majd létrehozom köztük a kapcsolatot. Az assertekben ellenőrzöm, hogy megfelelő-e a köztük lévő kapcsolat.



ábra 5.1. Két példa teszt, ami a modellek közti kapcsolatokat ellenőrzi

## Controller tests

4 fő osztályom van:

* AppStatusControllerTest
* QrCodeControllerTest
* WorkSpaceInvitationControllerTest
* WorkSpaceControllerTest

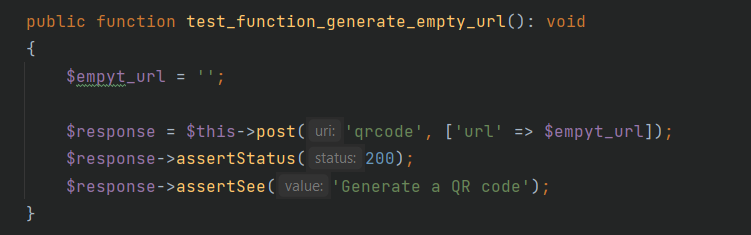
Mindegyik modellemnek az általános funkciói:

* index()
* show()
* update()
* edit()
* destroy()



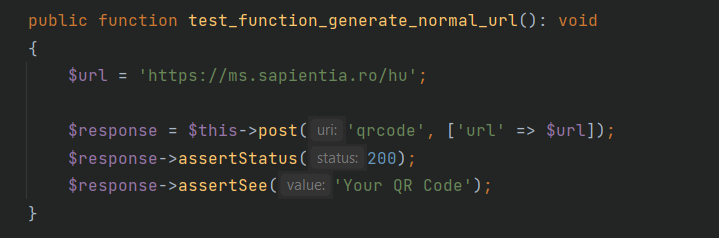
ábra 5.2. AppstatusControllerTest

Az (ábra 5.2) ábrán látható tesztben létrehozok egy felhasználót, ennek a felhasználónak egy munkateret, és egy appstatust, ami a munkatérhez tartozik. Az actingAs(user) a login műveletet végzi el automatikusan, hogy a tesztben a felhasználó már bejelentkezve érje el az adott oldalt. Az assertSee-vel ellenőrzöm, hogy az oldalon megjelent-e az azelőtt létrehozott egyedi appstatus.



ábra 5.3. Empty Qrcode Generation Test

Az oldal akkor működik helyesen, ha a felhasználó nem adott, meg semmit a QRcode generálás mezőben, akkor maradjon ugyan azon az oldalon. A ‘Generate a QR code’ felirat csak a kezdő oldalon látható, tehát ha a teszt továbbra is lássa a feliratot az oldalon, akkor helyes a működés.



ábra 5.4. Qrcode Generation Test

Az (ábra 5.4) ábrán látható tesztben egy valós url cím generálást tesztel a kód. A ‘Your QR Code’ felirat csak a kigenerált QR kód mellett látható. Amennyiben a 200-as státuszt kapunk vissza és látható a felirat, akkor a teszt sikeresnek bizonyult.

# Feature tests

A feature tesztelés a Laravel keretrendszerben egy olyan tesztelési folyamat, amely során a különböző funkciókat, illetve jellemzőket teszteljük, amelyeket a felhasználók valósítanak meg. A feature tesztek célja, hogy biztosítsák a Laravel alkalmazás helyes működését és végfelhasználóbarát legyen.

Itt két fő funkciót ellenőriztem:

* Login
* Register



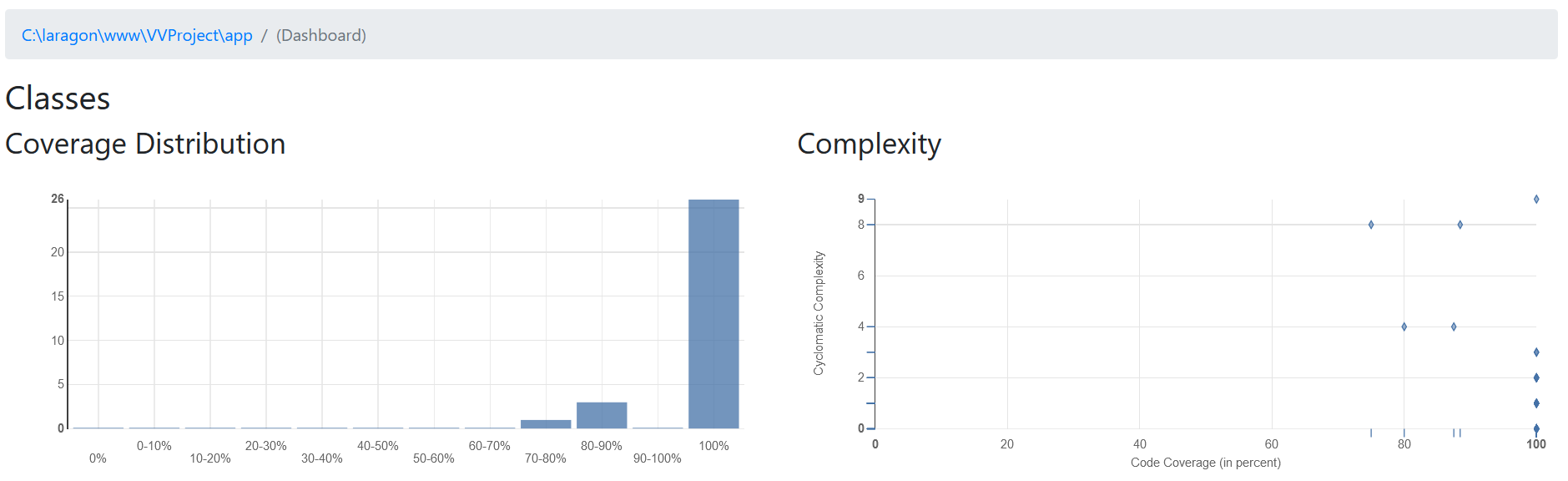
ábra 6.1. Login funkció tesztelése

A (ábra 6.1)-n látható **test\_login\_screen\_can\_be\_rendered()** tesztben azt ellenőrzöm, hogy a login oldal betöltődik-e? A teszt elnavigál a ‘/login’ oldalra. Ha 200-as választ kap, átment a teszten.

A **test\_user\_can\_authenticate\_using\_the\_login\_sreen()** – ben létrehozok egy usert, a korábban megírt factory segítségével. Ennek a usernek az email és jelszó kombinációjával megpróbálok bejelentkezni. Az assert atuhenticated ellenőrzi, hogy megfelelően bejelntkezett-e a user. Az assertRedirect ellenőrzi, hogy a sikeres bejelentkezés után a RouteServiceproviderben meghatározott helyre navigált-e az oldal.

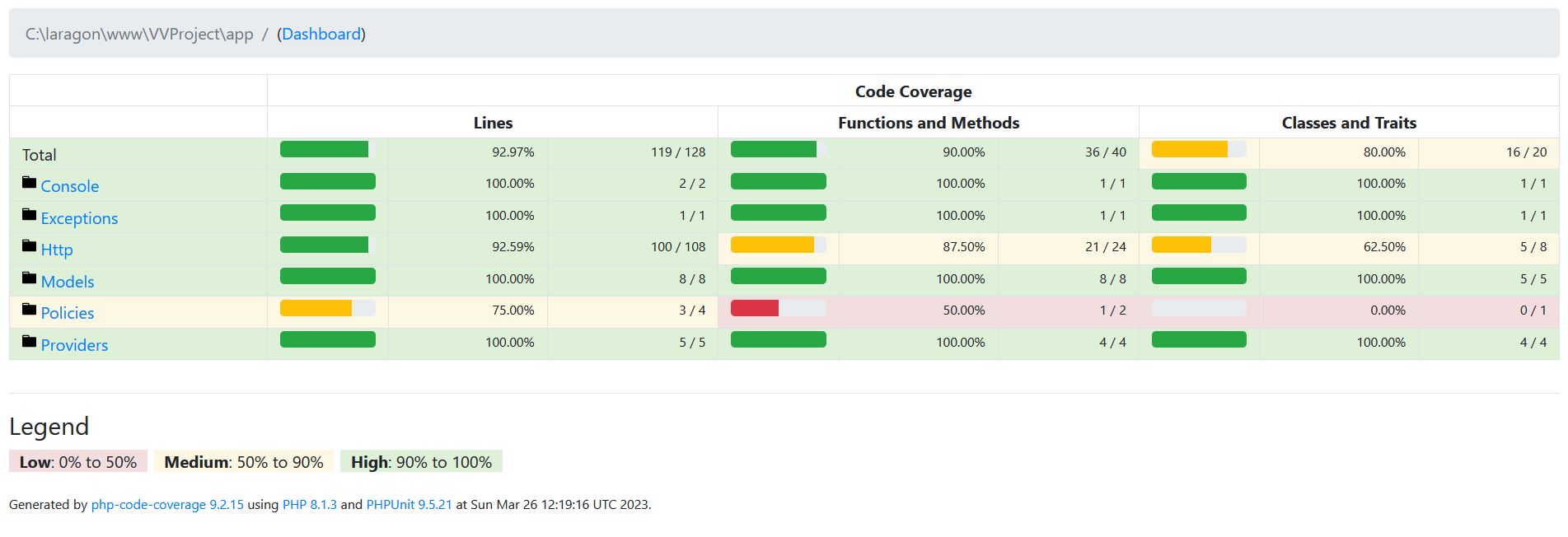
# Teszt eredmények

A Laravel keretrendszer tesztelési funkciói mellé letölthető egy bővítmény, xDebug. Ennek a segítségével egy reportot, jelentést lehet generáltatni automatikusan a lefuttatott tesztekről. Ezekrő különböző statisztikákat készít, minden fájlnak mutatja részletesen a kódlefedettségét.



ábra 7.1. Teszt lefedettség eloszlása

A (ábra 7.1) ábrán látható, hogy a legtöbb teszt 100%-s lefedettséget nyújt, lefedi az adott kódrész 100%át. Összesen 35 teszt van, az összes sikeresen lefut.



ábra 7.2. Kód lefedettség részletesen, egységenként

Az (ábra 7.2)-n látható részletesebben az összes mappa kód lefedettsége. Az eredeti program tartalmaz egy Admin mappát is, ahol az adminisztrátori joggal rendelkező felhasználókkal kapcsolatos funkciók találhatóak. Mivel ez még nem került implementálásra, nem volt konkrét funkciója, ezért azt kivettem a code coverage-ből. A tesztekről további részletek találhatóak a repostitory[6] mappájában.

# Irodalomjegyzék

1. <https://laravel.com/docs/10.x/installation>
2. <https://medium.com/@anowarhossain/code-coverage-report-in-laravel-and-make-100-coverage-of-your-code-ce27cccbc738>
3. <https://stackoverflow.com/questions/14379569/force-exclude-files-from-phpunit-code-coverage>
4. <https://laravel.com/docs/10.x/testing>
5. <https://laragon.org/>
6. <https://github.com/szabilukacs/VVProject/tree/main/tests/Coverage/html>