

UNIwersytet w Białymstoku
Wydział Ekonomiczno-Informatyczny w Wilnie



Sprawozdanie 2

Laravel REST API

Damian Rusakiewicz

Programowanie we frameworkach

Informatyka, 3 rok

Wilno 2022

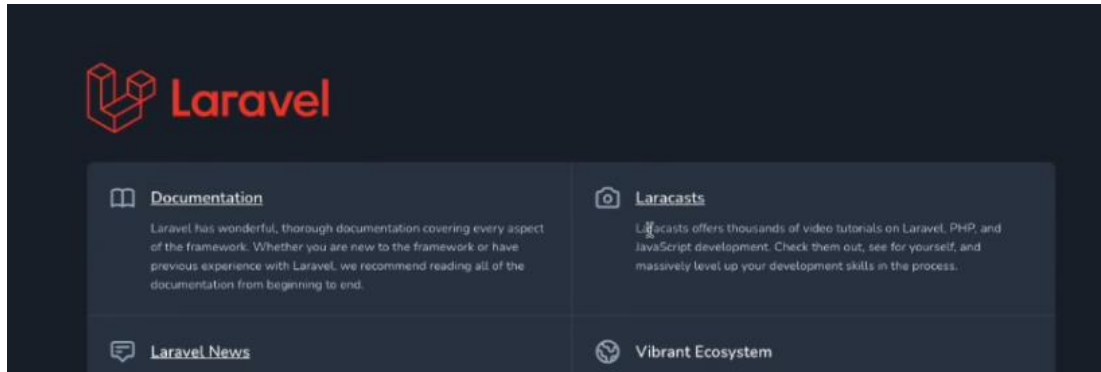
| | | |
|-----|---|---------------------|
| 1. | Setting Up | 2 |
| 2. | SQLite Setup | 2 |
| 3. | Model | 4-5 |
| 4. | Update Product | 5-6 |
| 5. | Delete Product | 6 |
| 6. | Search Products | 6 |
| 7. | Auth Controller | 7 |
| 8. | Zarejestruj użytkownika i uzyskaj token | 7 |
| 9. | Token | 8 |
| 10. | Register | 8 |
| 11. | Body | 8 |
| 12. | Pre-Request Script | 8 |
| 13. | Logout i Delete | 9 |
| 14. | Login User i Get Token | 10 |
| 15. | Link do kodu | 10 |

Setting Up.

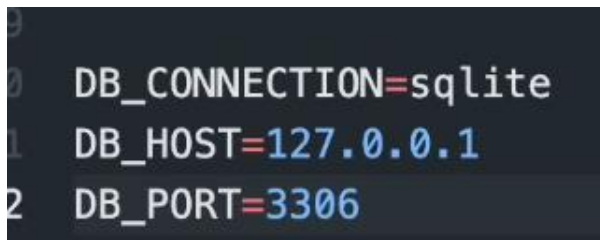
Najpierw tworzę nowy projekt z kompozytorem.

Następnie włączam aplikację.

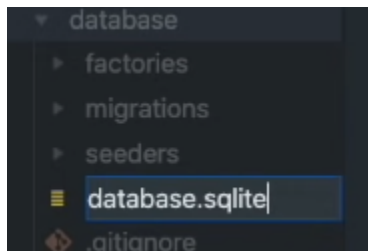
Tak wygląda strona internetowa.



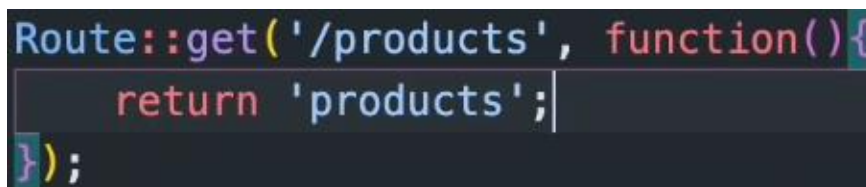
SQLite Setup.



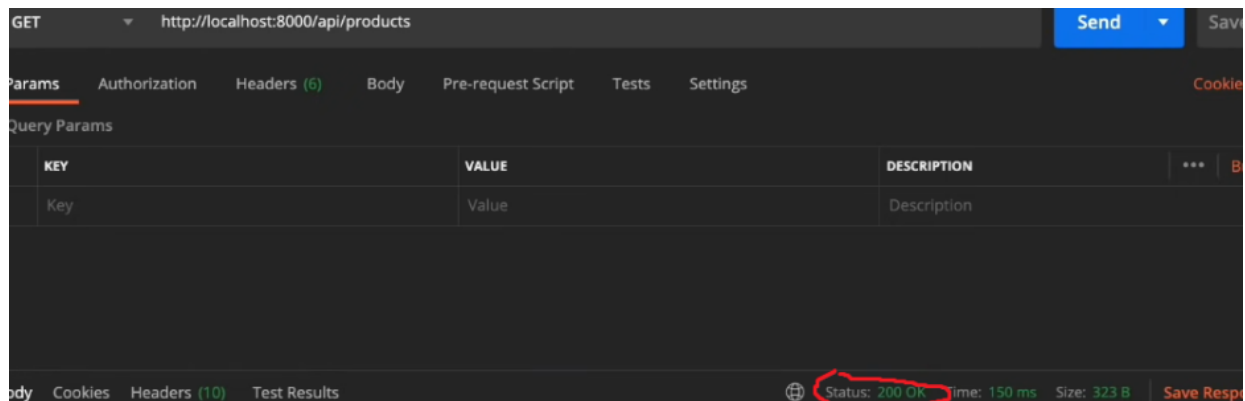
Następnie tworzę plik bazy danych.



Route api.php.



Po dodaniu kodu w api.php, tak wygląda nasz Sqlite. Widzimy, że to działa, bo dostajemy **Status: 200 OK**.



Teraz tworzę modele dla naszych produktów.

```
php artisan make:model Product --migration
```

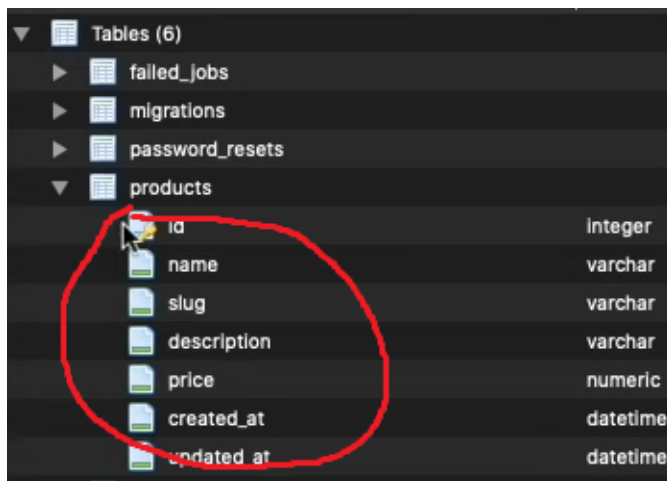
Następnie dodaję tables.

```
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('slug');
        $table->string('description')->nullable();
        $table->decimal('price', 5, 2);
        $table->timestamps();
    });
}
```

Teraz migracje są gotowe do uruchomienia.

```
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (1.98ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (1.56ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
```

Widzimy, że zostały one dodane do bazy danych.



| Tables (6) | |
|-----------------|----------|
| failed_jobs | |
| migrations | |
| password_resets | |
| products | |
| id | integer |
| name | varchar |
| slug | varchar |
| description | varchar |
| price | numeric |
| created_at | datetime |
| updated_at | datetime |

Model.


```
Route::post('/products', function() {  
    return Product::create([  
        'name' => 'Product One',  
        'slug' => 'product-one',  
        'description' => 'This is product one',  
        'price' => '99.99'  
    ]);  
});
```

```
"name": "Product One",  
"slug": "product-one",  
"description": "This is product one",  
"price": "99.99",  
"updated_at": "2021-03-30T20:39:00.000000Z",  
"created_at": "2021-03-30T20:39:00.000000Z",  
"id": 1
```

| | id | name | slug | description | price |
|-----|--------|-------------|-------------|---------------------|--------|
| ... | Filter | Filter | Filter | Filter | Filter |
| 1 | 1 | Product One | product-one | This is product one | 99.99 |

Tworzenie produktów i walidacja.

Params Authorization **Headers (8)** Body Pre-request Script Test

Headers  7 hidden

| | KEY | VALUE |
|-------------------------------------|--------|------------------|
| <input checked="" type="checkbox"/> | Accept | application/json |

Params Authorization Headers (9) **Body** Pre-request Script

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

| | KEY | VALUE |
|-------------------------------------|-------------|---------------------|
| <input checked="" type="checkbox"/> | name | Product Two |
| <input checked="" type="checkbox"/> | slug | product-two |
| <input checked="" type="checkbox"/> | description | This is product two |
| <input checked="" type="checkbox"/> | price | 299.99 |

```
{
  "name": "Product Two",
  "slug": "product-two",
  "description": "This is product two",
  "price": "299.99",
  "updated_at": "2021-03-30T20:45:51.000000Z",
  "created_at": "2021-03-30T20:45:51.000000Z",
  "id": 2
}
```

Update Product.

```
public function update(Request $request, $id)
{
    $product = Product::find($id);
    $product->update($request->all());
    return $product;
}
```

Updated cenę, aby sprawdzić, czy to działa.

```
{
  "id": 2,
  "name": "Product Two",
  "slug": "product-two",
  "description": "This is product two",
  "price": "199.99",
  "created_at": "2021-03-30T20:45:51.000000Z",
  "updated_at": "2021-03-30T20:51:36.000000Z"
}
```

Delete Product.

```
public function destroy($id)
{
    return Product::destroy($id);
}
```

Produkt 3 został usunięty.

DELETE <http://localhost:8000/api/products/3> Send

Search Products.

```
/**
 * Search for a name
 *
 * @param str $name
 * @return \Illuminate\Http\Response
 */
public function search($name)
{
    return Product::where('name', 'like', '%'.$name.'%')->get();
}
```

Stwarzam Route.

```
Route::resource('products', ProductController::class);
Route::get('/products/search/{name}', [ProductController::class, 'search']);
```


Teraz wpisuję wyszukiwanie "iph".



Search pracuje dobrze (Znalazł nazwę iPhone na podstawie samego „iph”).

```
{
  "id": 4,
  "name": "iPhone 12",
  "slug": "iphone-12",
  "description": "This is the iPhone 12",
  "price": "599.99",
  "created_at": "2021-03-30T20:55:18.000000Z",
}
```

Auth Controller.

```
php artisan make:controller AuthController
```

Zarejestruj użytkownika i uzyskaj token.

```
class AuthController extends Controller
{
    public function register(Request $request) {
        $fields = $request->validate([
            'name' => 'required|string',
            'email' => 'required|string|unique:users,email',
            'password' => 'required|string|confirmed'
        ]);

        $user = User::create([
            'name' => $fields['name'],
            'email' => $fields['email'],
            'password' => bcrypt($fields['password'])
        ]);
    }
}
```


Token.

```
$token = $user->createToken('myapptoken')->plainTextToken;


$response = [
    'user' => $user,
    'token' => $token
];

return response($response, 201);
```

Register.

```
Route::post('/register', [AuthController::class, 'register']);
```

Body.

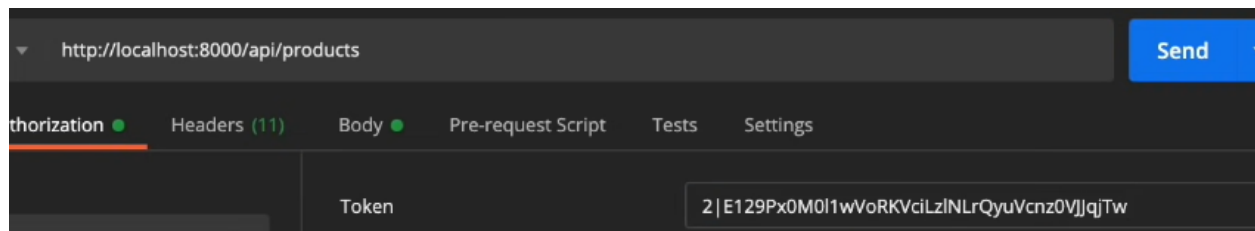
| Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings |
|--|---------------|------------------|------|--------------------|-------|----------|
| Headers  8 hidden | | | | | | |
| | KEY | VALUE | | | | |
| <input checked="" type="checkbox"/> | Accept | application/json | | | | |

Pre-Request Script.

| Params | Authorization | Headers (10) | Body ● | Pre-request Script | Tests | Settings |
|---|-----------------------|----------------|--------|--------------------|-------|----------|
| ● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL | | | | | | |
| | KEY | VALUE | | | | |
| <input checked="" type="checkbox"/> | name | Brad | | | | |
| <input checked="" type="checkbox"/> | email | brad@gmail.com | | | | |
| <input checked="" type="checkbox"/> | password | 123456 | | | | |
| <input checked="" type="checkbox"/> | password_confirmation | 123456 | | | | |

Po rejestracji dostałem token

```
{
  "token": "2|E129Px0M01wYpRKVciLz1NLRQyuVcnz0VJJqjTw"
```



Po kliknięciu Send

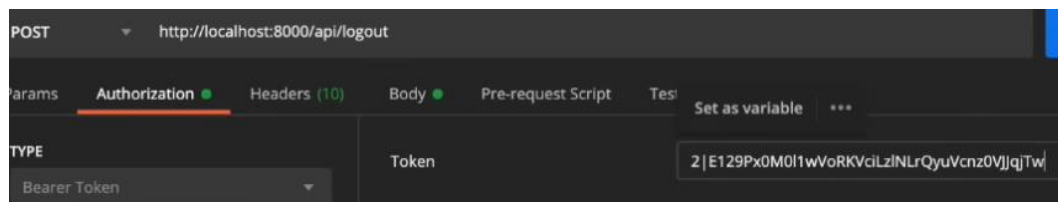
```
"name": "Test Product",
"slug": "test-product",
"description": "This is a test",
"price": "599.99",
"updated_at": "2021-03-30T21:31:48.000000Z",
"created_at": "2021-03-30T21:31:48.000000Z",
"id": 5
```

Logout i Delete.

```
public function logout(Request $request) {
    auth()->user()->tokens()->delete();

    return [
        'message' => 'Logged out'
    ];
}
```

```
Route::post('/logout', [AuthController::class, 'logout']);
```



```
{
  "message": "Logged out"
}
```

Login User i Get Token.

```
// Check email  
$user = User::where('email', $fields['email'])->first();
```

```
// Check password  
if(!user || !Hash::check($fields['password'], $user->password)) {  
    return response([  
        'message' => 'Bad creds'  
    ], 401);  
}
```

Login Spracowal.

```
{  
  "token": "3|L5y41TiyLjWeDgSAybTdaGwMSFBnp8d1SRUDPupG"
```

Link do kodu:

<https://drive.google.com/drive/folders/11AYt8BJhwNMv0Rmr8w4TcAdwtAe8675p>