

**Ministerul Educației, Culturii și Cercetării a Republicii Moldova**

**Universitatea de Stat a Moldovei**

**Facultatea Matematică și Informatică**

Raport

**Lucrare de laborator Nr.5  
la disciplina Framework-uri pentru Dezvoltarea de Aplicații Web  
Tema:”** **Baze de date și doctrină ORM”**

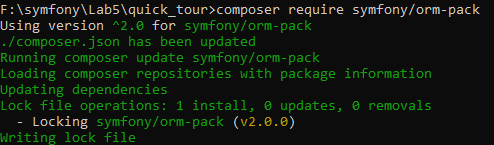
**A efectuat : Batîr Ana**

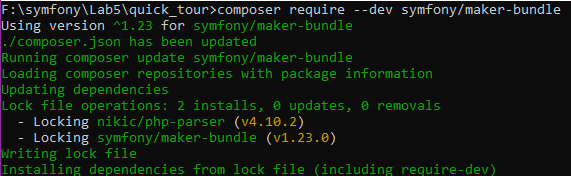
**A verificat : Bodrug Svetlana**

**Chișinău 2020**

# **Instalarea Doctrinei**

Mai întâi, instalez suport Doctrine prin pachetul orm Symfony, precum și MakerBundle, care mă va ajuta să generez un anumit cod:

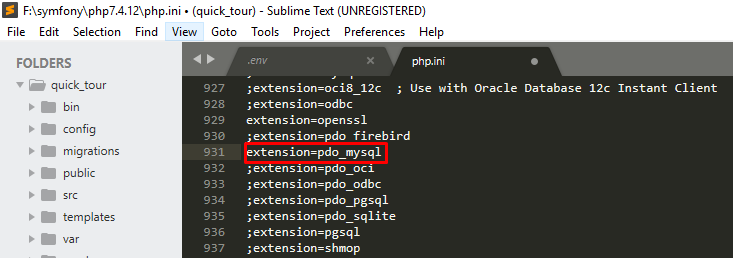




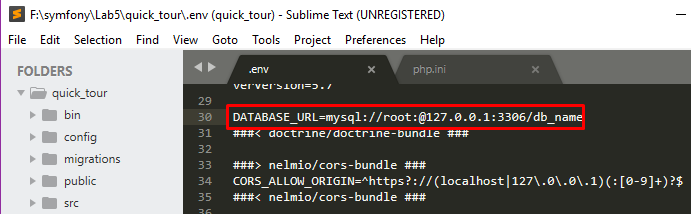
# **Configurarea bazei de date**

Informațiile de conectare la baza de date sunt stocate ca o variabilă de mediu numită DATABASE\_URL. Pentru dezvoltare, pot găsi și personaliza acest lucru în interiorul .env.

Activare MySQL în fișierul php.ini .



Acum că parametrii de conexiune sunt configurați, Doctrine poate crea baza de date db\_name:

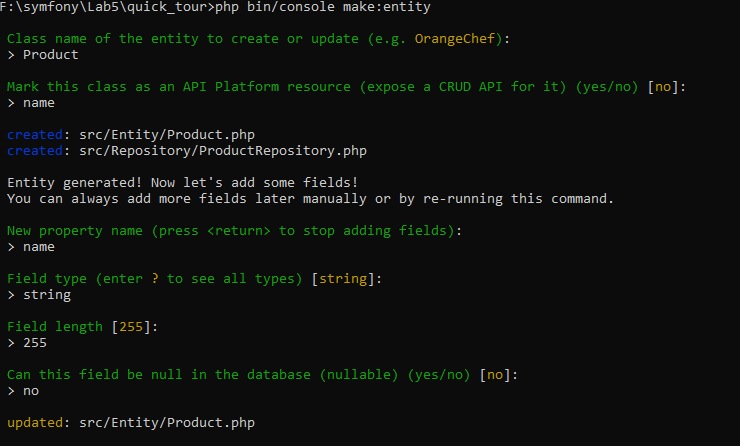




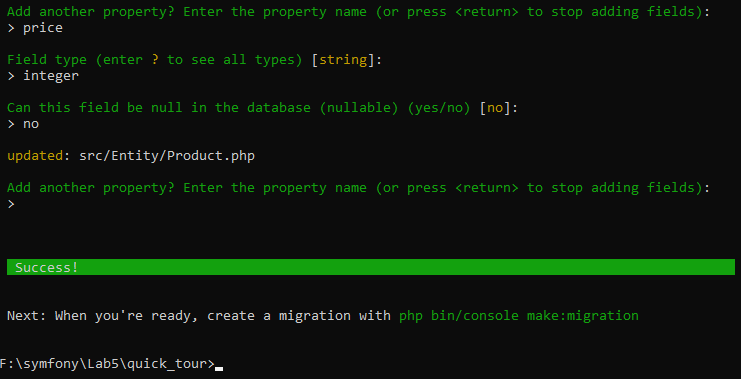
# **Crearea unei clase de entitate**

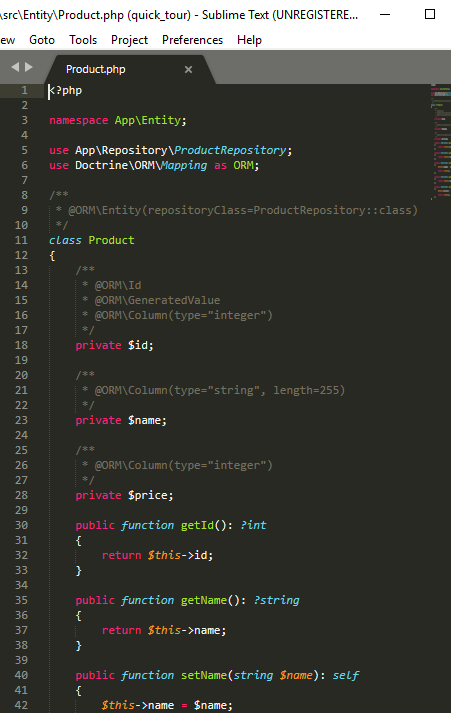
Să presupunem că creez o aplicație în care trebuie afișate produse. Fără să vă gândiți la Doctrină sau baze de date, știm deja că am nevoie de un obiect Produs pentru a reprezenta aceste produse.

Pot utiliza comanda make: entity pentru a crea această clasă și orice câmpuri de care am nevoie. Comanda va pune câteva întrebări - răspund așa cum se face mai jos:



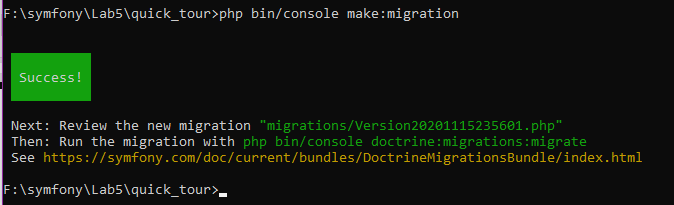
Iuhu! Acum am un nou fișier src / Entity / Product.php:



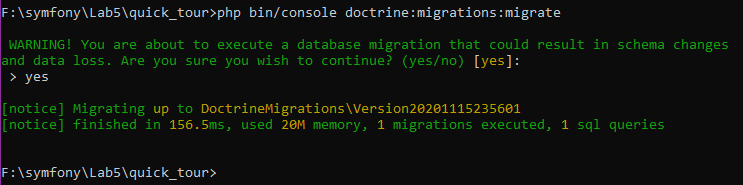


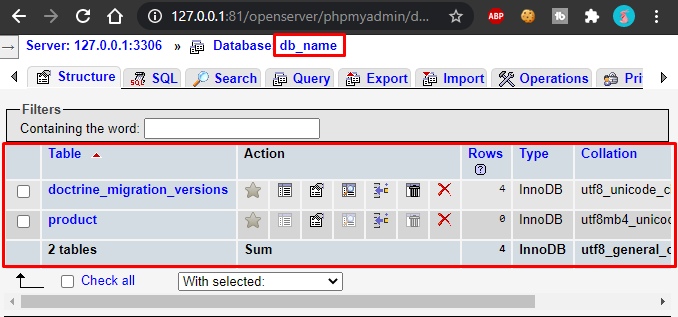
# **Migrații: crearea tabelelor / schemei bazei de date**

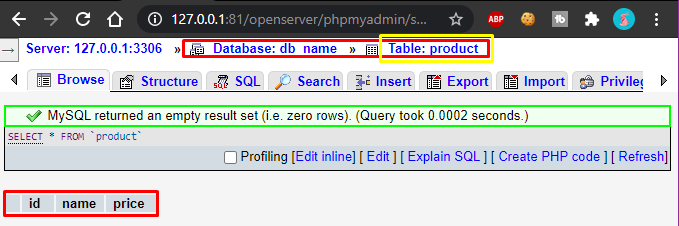
Clasa de produs este complet configurată și gata de salvare într-un tabel de produse. Dacă tocmai am definit această clasă, baza de date nu are încă tabelul de produse. Pentru a-l adăuga, pot utiliza DoctrineMigrationsBundle, care este deja instalat:



Dacă deschid acest fișier, acesta conține codul SQL necesar pentru actualizarea bazei de date! Pentru a rula acel SQL, execut migrările:

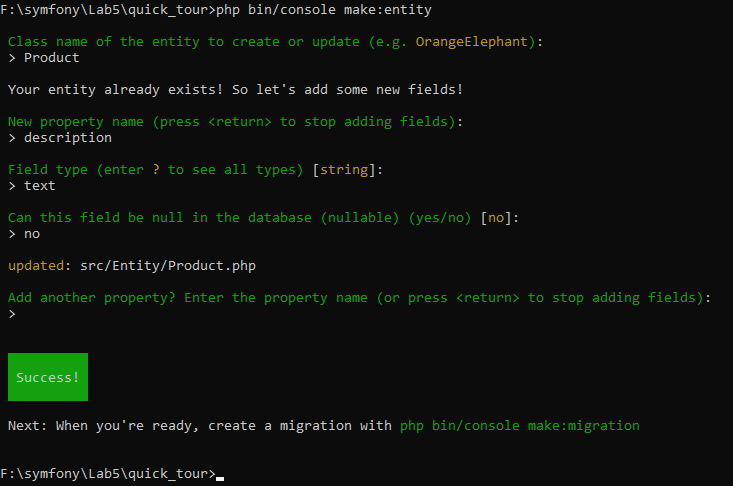


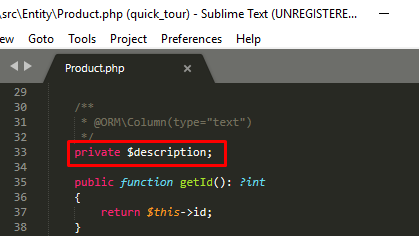




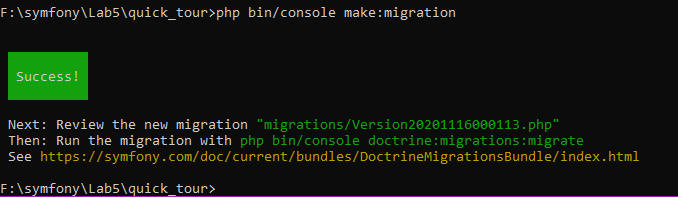
# **Migrații și adăugarea mai multor câmpuri**

Dar dacă trebuie să adaug o nouă proprietate de câmp la produs, cum ar fi o descriere? Pot edita clasa pentru a adăuga noua proprietate. Dar, pot utiliza și make: entity din nou:

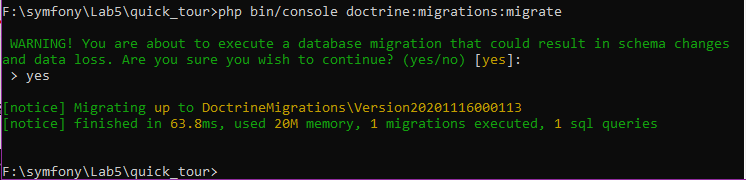


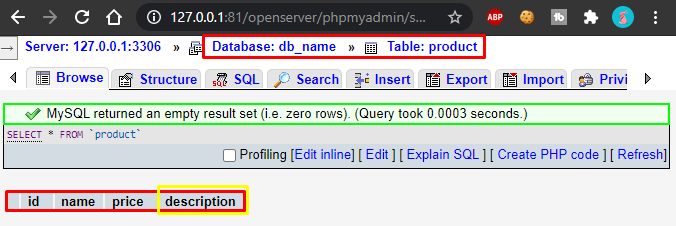


Noua proprietate este mapată, dar nu există încă în tabelul de produse. Nici o problema! Generez o nouă migrație:



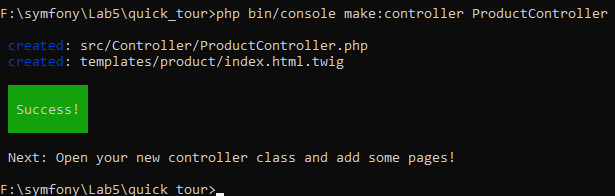
Sistemul de migrație este inteligent. Compară toate entitățile cu starea actuală a bazei de date și generează SQL-ul necesar pentru a le sincroniza! Ca și înainte, execut migrările:



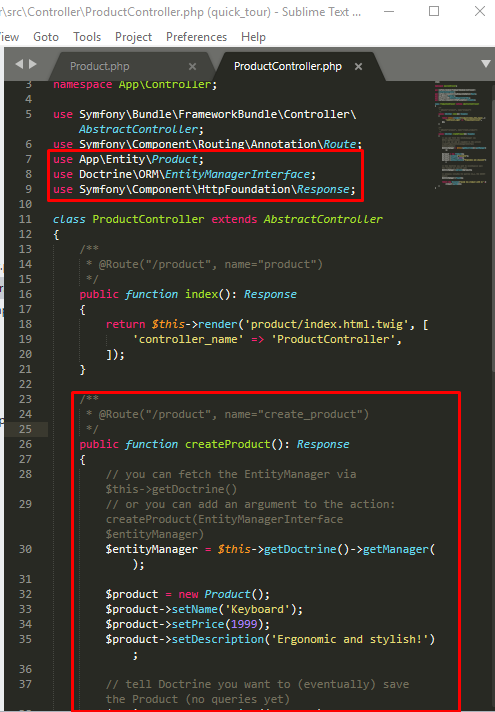


# **Obiecte persistente în baza de date**

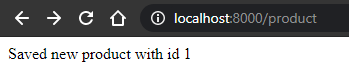
Este timpul să salvez un obiect Produs în baza de date! Să creez un controler nou pentru a experimenta:

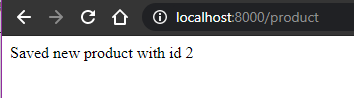


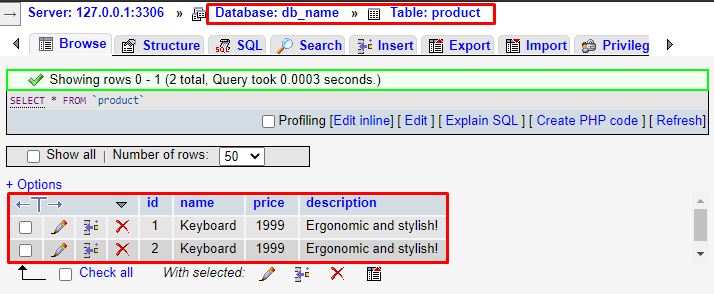
În interiorul controlerului, pot crea un nou obiect de produs, pot seta date pe acesta și îl pot salva:



**REZULTAT:**







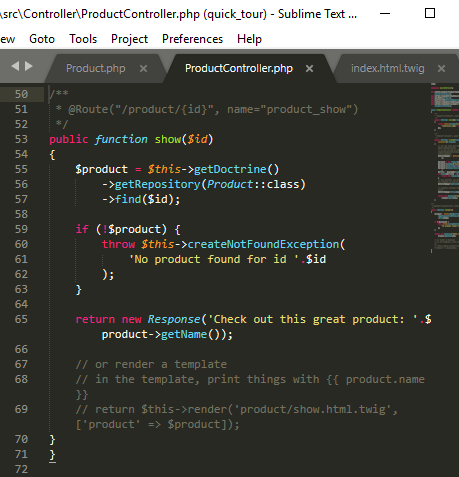
# **Validarea obiectelor**

Validatorul Symfony reutilizează metadatele Doctrine pentru a efectua câteva sarcini de validare de bază:

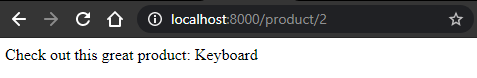


# **Preluarea obiectelor din baza de date**

Extragerea unui obiect din baza de date este și mai ușoară. Să presupunem că pot merge la / product / 1 pentru a vedea noul produs:

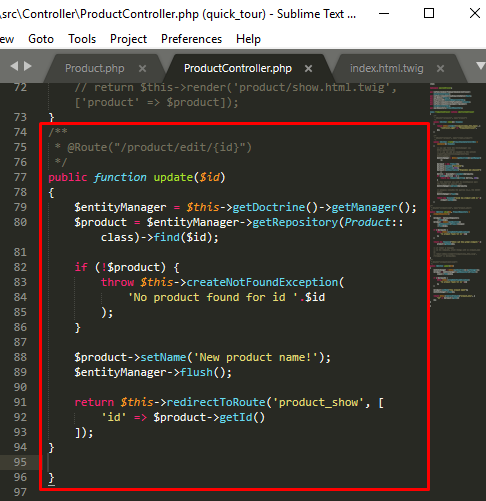


**REZULTAT:**



# **Actualizarea unui obiect**

După ce am preluat un obiect din Doctrină, interacționez cu el la fel ca cu orice model PHP:



**REZULTAT:**

