BLUE SKY DIGITAL LABS

TASK 1

Parakram Vishwakarma | GitHub link: https://github.com/Parakramvishwakarma/BlueSky_Task1

TABLE OF CONTENTS

INTRODUCTION

This task covered the basic environment of PHP server side. Composer a PHP project-based dependency manager is used to create a PHP project which is an application with the symphony framework. Using Symfony we have access to multiple libraries which allows the application created to communicate with the database using the console.

MySQL server was used to connect to the database along with the apache server which were both received from the use of XAMPP. The application created is a "database abstraction layer" that nullifies the need of php scripts to create databases and interact with them. The functions can be achieved from the console through the help of the DOCTRINE ORM which is installed through the symfony component 'orm-pack'.

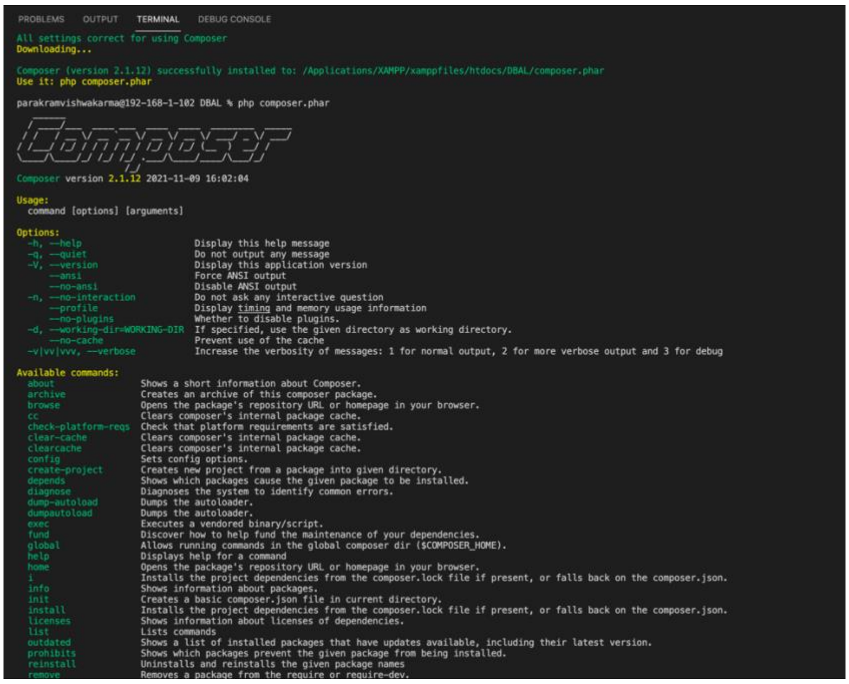
The DBAL created can perform SQL Queries from the command line through the command

`doctrine:query:sql` *`"SQL QUERY DESIRED"`*

METHOD

COMPOSER DOWNLOAD

The composer is downloaded from the bash script available on the website and `composer.phar` is put in a directory on the PATH to make it globally accessible.



```
All settings correct for using Composer
Downloading...

Composer (version 2.1.12) successfully installed to: /Applications/XAMPP/xamppfiles/htdocs/DBAL/composer.phar
Use it: php composer.phar

parakramvishwakarma@192-168-1-102 DBAL % php composer.phar

Composer version 2.1.12 2021-11-09 16:02:04

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directory.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  about                Shows a short information about Composer.
  archive              Creates an archive of this composer package.
  browse               Opens the package's repository URL or homepage in your browser.
  cc                   Clears composer's internal package cache.
  check-platform-reqs  Check that platform requirements are satisfied.
  clear-cache          Clears composer's internal package cache.
  clearcache           Clears composer's internal package cache.
  config               Sets config options.
  create-project       Creates new project from a package into given directory.
  depends              Shows which packages cause the given package to be installed.
  diagnose             Diagnoses the system to identify common errors.
  dump-autoload        Dumps the autoloader.
  dumpautoload         Dumps the autoloader.
  exec                 Executes a vendored binary/script.
  fund                 Discover how to help fund the maintenance of your dependencies.
  global               Allows running commands in the global composer dir ($COMPOSER_HOME).
  help                 Displays help for a command
  home                 Opens the package's repository URL or homepage in your browser.
  i                    Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json.
  info                 Shows information about packages.
  init                 Creates a basic composer.json file in current directory.
  install              Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json.
  licenses             Shows information about licenses of dependencies.
  list                 Lists commands
  outdated             Shows a list of installed packages that have updates available, including their latest version.
  prohibits            Shows which packages prevent the given package from being installed.
  reinstall            Uninstalls and reinstalls the given package names
  remove               Removes a package from the require or require-dev.
```

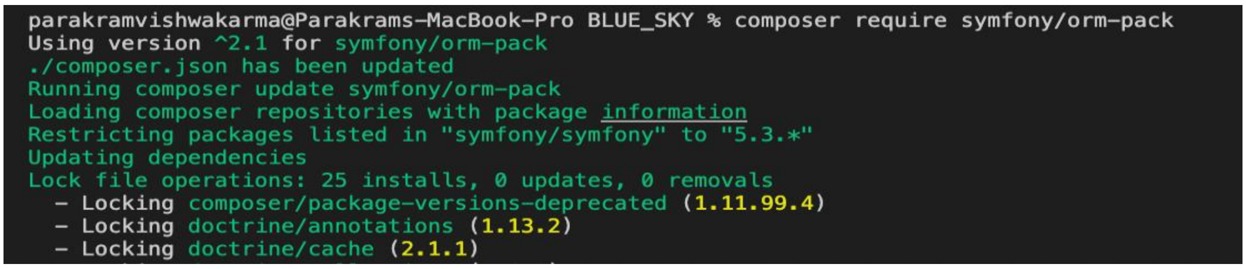
CREATING THE SYMFONY APPLICATION

The permission inside the htdocs of the XAMPP/xamppfiles was changed to 755 and then a new Symfony project was created inside htdocs directory using the command below.

```
Composer create-project symphony/skeleton BLUE_SKY
```

ADDING DOCTRINE ORM

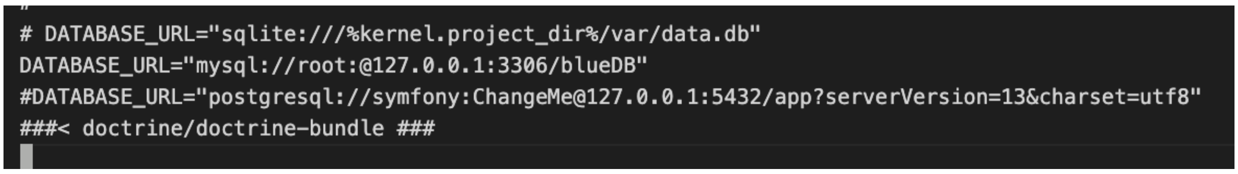
Then the orm-pack symphony component was added to the project:



```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % composer require symfony/orm-pack
Using version ^2.1 for symfony/orm-pack
./composer.json has been updated
Running composer update symfony/orm-pack
Loading composer repositories with package information
Restricting packages listed in "symfony/symfony" to "5.3.*"
Updating dependencies
Lock file operations: 25 installs, 0 updates, 0 removals
  - Locking composer/package-versions-deprecated (1.11.99.4)
  - Locking doctrine/annotations (1.13.2)
  - Locking doctrine/cache (2.1.1)
```

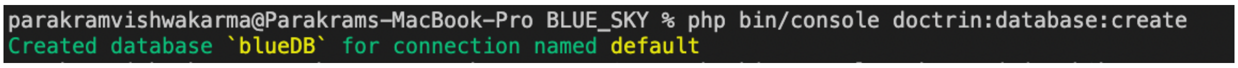
CREATING THE DATABASE USING DOCTRINE

The DATABASE_URL configurations were then updated in the .env file to add the username and password for the MySQL server and the name of the database being created was added (named 'blueDB').



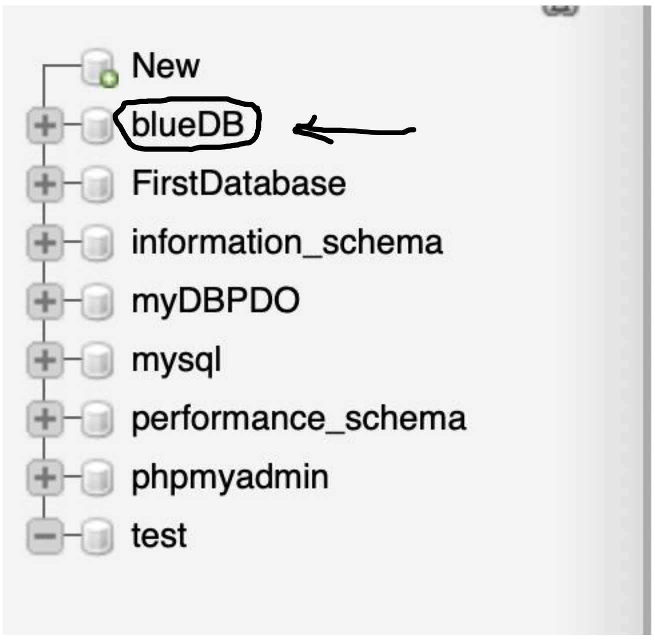
```
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
DATABASE_URL="mysql://root:@127.0.0.1:3306/blueDB"
#DATABASE_URL="postgresql://symfony:ChangeMe@127.0.0.1:5432/app?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###
```

The database was then created using Doctrine.



```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console doctrin:database:create
Created database `blueDB` for connection named default
```

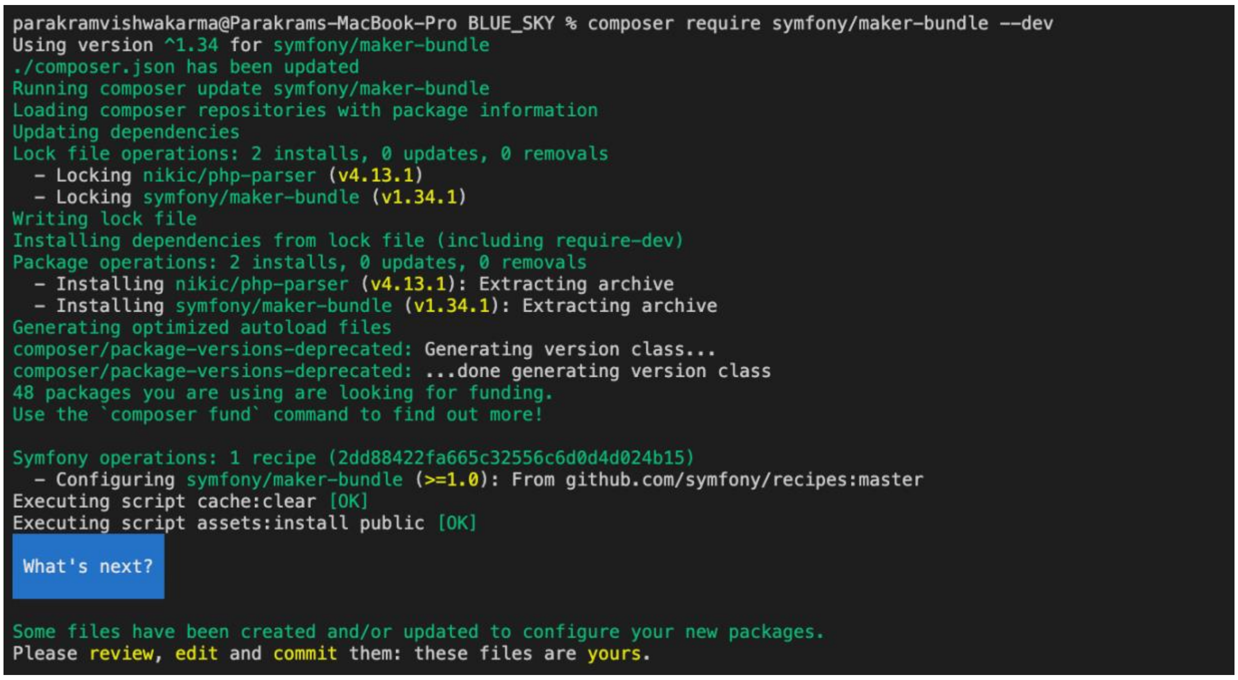
The creation of the database was checked through the phpMyAdmin GUI for databases:



SUCCESS!

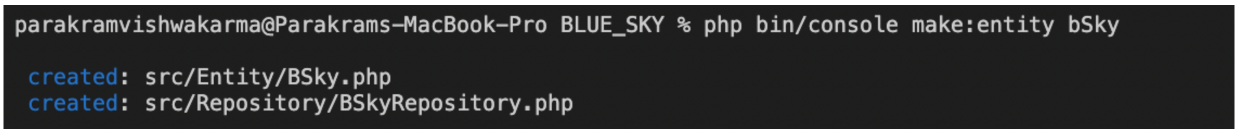
ADDING A TABLE

The MakerBundle component was then required to get the "make" command to create entities.



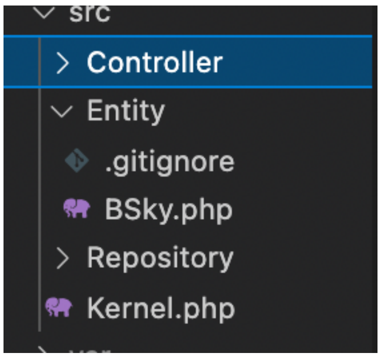
```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % composer require symfony/maker-bundle --dev
Using version ^1.34 for symfony/maker-bundle
./composer.json has been updated
Running composer update symfony/maker-bundle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
  - Locking nikic/php-parser (v4.13.1)
  - Locking symfony/maker-bundle (v1.34.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
  - Installing nikic/php-parser (v4.13.1): Extracting archive
  - Installing symfony/maker-bundle (v1.34.1): Extracting archive
Generating optimized autoload files
composer/package-versions-deprecated: Generating version class...
composer/package-versions-deprecated: ...done generating version class
48 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Symfony operations: 1 recipe (2dd88422fa665c32556c6d0d4d024b15)
  - Configuring symfony/maker-bundle (>=1.0): From github.com/symfony/recipes:master
Executing script cache:clear [OK]
Executing script assets:install public [OK]

 What's next?

Some files have been created and/or updated to configure your new packages.
Please review, edit and commit them: these files are yours.
```

A table was then inserted into the database called ‘bSky’.

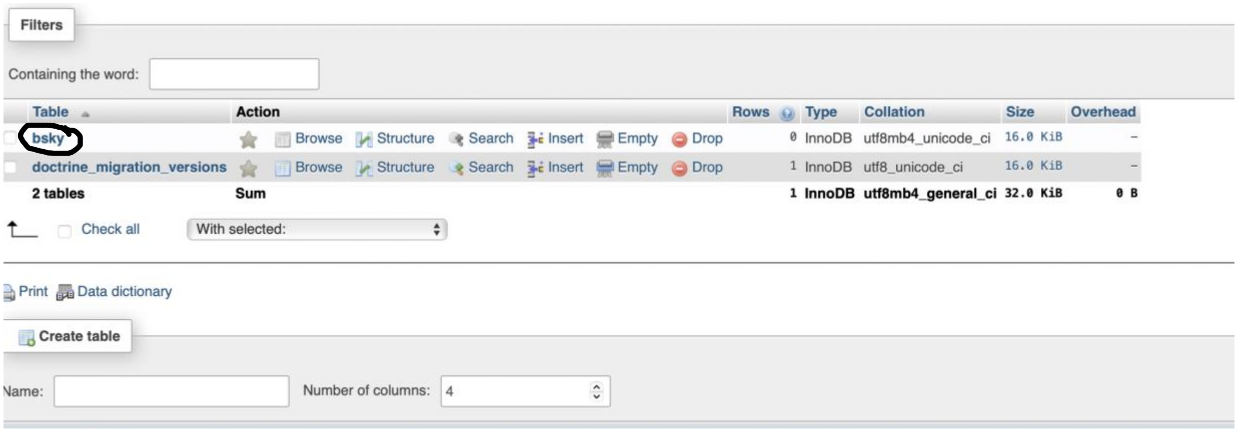


```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console make:entity bSky

 created: src/Entity/BSky.php
 created: src/Repository/BSkyRepository.php
```

This was visible in the /src/Entitiy folder in the project directory



```
 ∨ src
   > Controller
   ∨ Entity
      .gitignore
      BSky.php
   > Repository
   Kernel.php
```



Filters

Containing the word:

Table	Action							Rows	Type	Collation	Size	Overhead
bsky	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
doctrine_migration_versions	☆	Browse	Structure	Search	Insert	Empty	Drop	1	InnoDB	utf8_unicode_ci	16.0 KiB	-
2 tables	Sum							1	InnoDB	utf8mb4_general_ci	32.0 KiB	0 B

Check all   With selected:

Print   Data dictionary

Create table

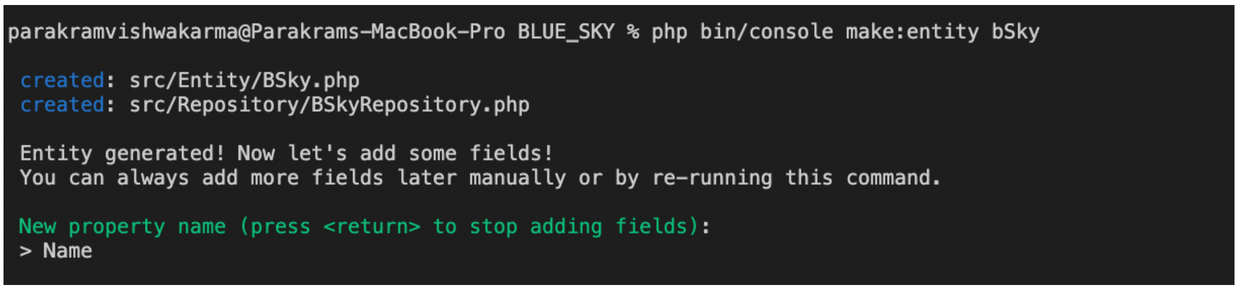
Name: Number of columns: 4

As we can see in the database the table is created along with a table to store the version of migrations.

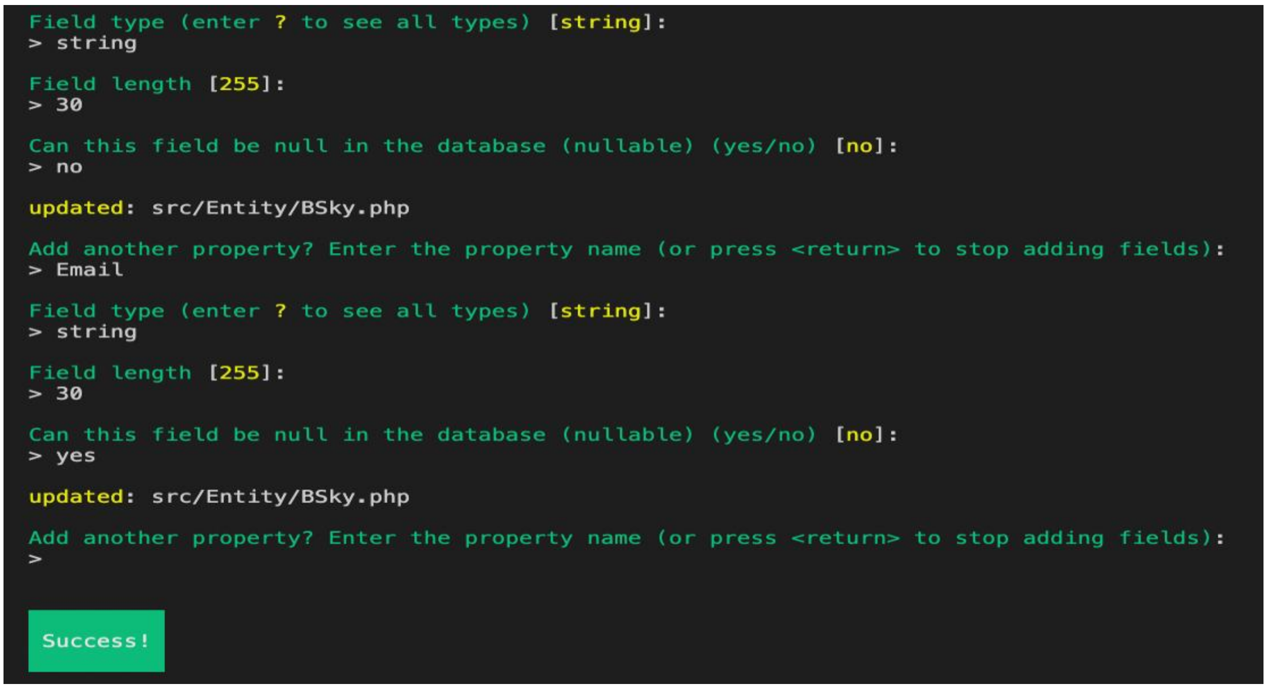
ADDING FIELDS

Since we have the maker bundle it is easy to define fields in the entity (table we just added).

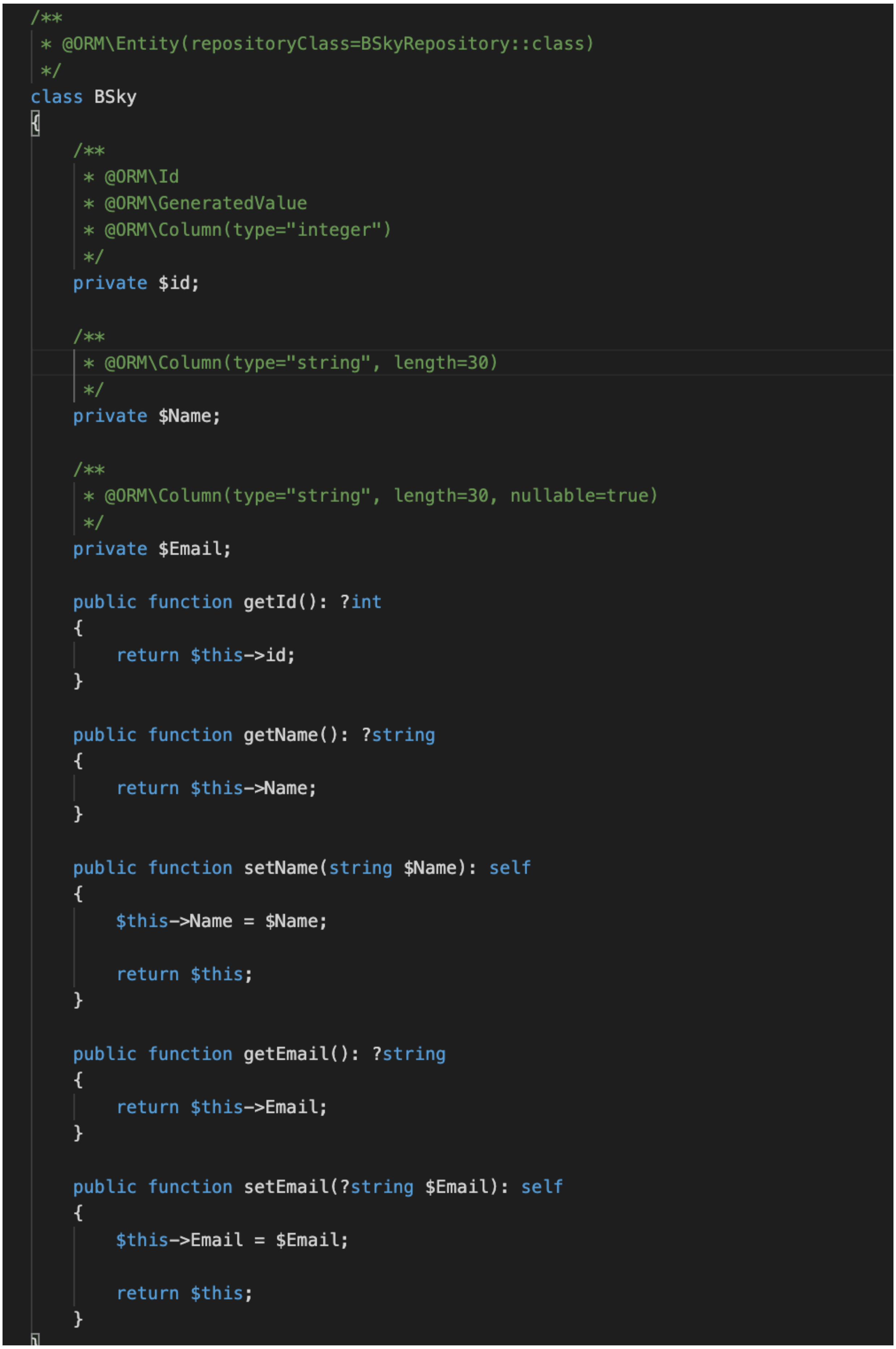
As soon as the table is created it, the ‘make’ command asks to add fields to the entity and these fields are added straight to the entity class in the src/Controller/Bsky.php



```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console make:entity bSky

 created: src/Entity/BSky.php
 created: src/Repository/BSkyRepository.php

 Entity generated! Now let's add some fields!
 You can always add more fields later manually or by re-running this command.

 New property name (press <return> to stop adding fields):
 > Name
```



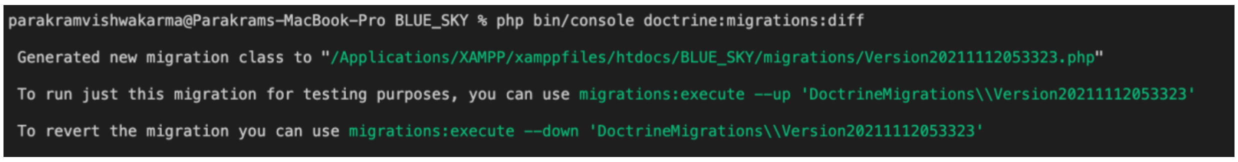
```
 Field type (enter ? to see all types) [string]:
 > string

 Field length [255]:
 > 30

 Can this field be null in the database (nullable) (yes/no) [no]:
 > no

 updated: src/Entity/BSky.php

 Add another property? Enter the property name (or press <return> to stop adding fields):
 > Email

 Field type (enter ? to see all types) [string]:
 > string

 Field length [255]:
 > 30

 Can this field be null in the database (nullable) (yes/no) [no]:
 > yes

 updated: src/Entity/BSky.php

 Add another property? Enter the property name (or press <return> to stop adding fields):
 >


 Success!
```



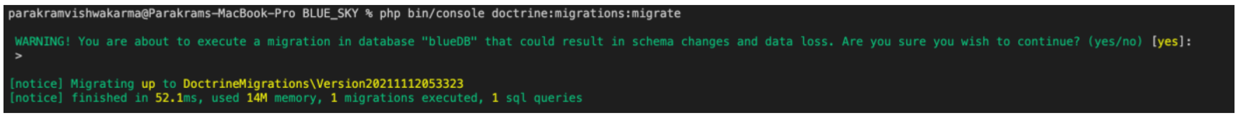
```php
/**
 * @ORM\Entity(repositoryClass=BSkyRepository::class)
 */
class BSky
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=30)
     */
    private $Name;

    /**
     * @ORM\Column(type="string", length=30, nullable=true)
     */
    private $Email;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getName(): ?string
    {
        return $this->Name;
    }

    public function setName(string $Name): self
    {
        $this->Name = $Name;

        return $this;
    }

    public function getEmail(): ?string
    {
        return $this->Email;
    }

    public function setEmail(?string $Email): self
    {
        $this->Email = $Email;

        return $this;
    }
}
```

We can add fields through the Bsky.php file in the src/Entity/.

The doctrine package is then used to first create a migration class and then use that to migrate the fields to their table

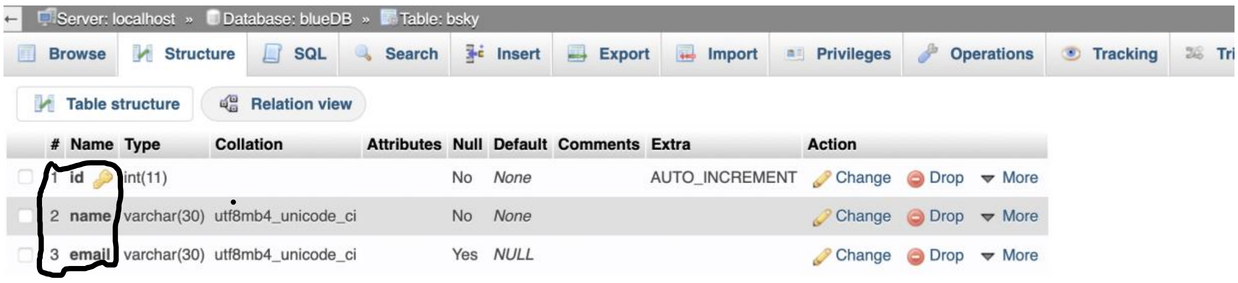


```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console doctrine:migrations:diff

 Generated new migration class to "/Applications/XAMPP/xamppfiles/htdocs/BLUE_SKY/migrations/Version20211112053323.php"

 To run just this migration for testing purposes, you can use migrations:execute --up 'DoctrineMigrations\\Version20211112053323'

 To revert the migration you can use migrations:execute --down 'DoctrineMigrations\\Version20211112053323'
```



```
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console doctrine:migrations:migrate

 WARNING! You are about to execute a migration in database "blueDB" that could result in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
 >

[notice] Migrating up to DoctrineMigrations\Version20211112053323
[notice] finished in 52.1ms, used 14M memory, 1 migrations executed, 1 sql queries
```

CHECK FOR FIELDS

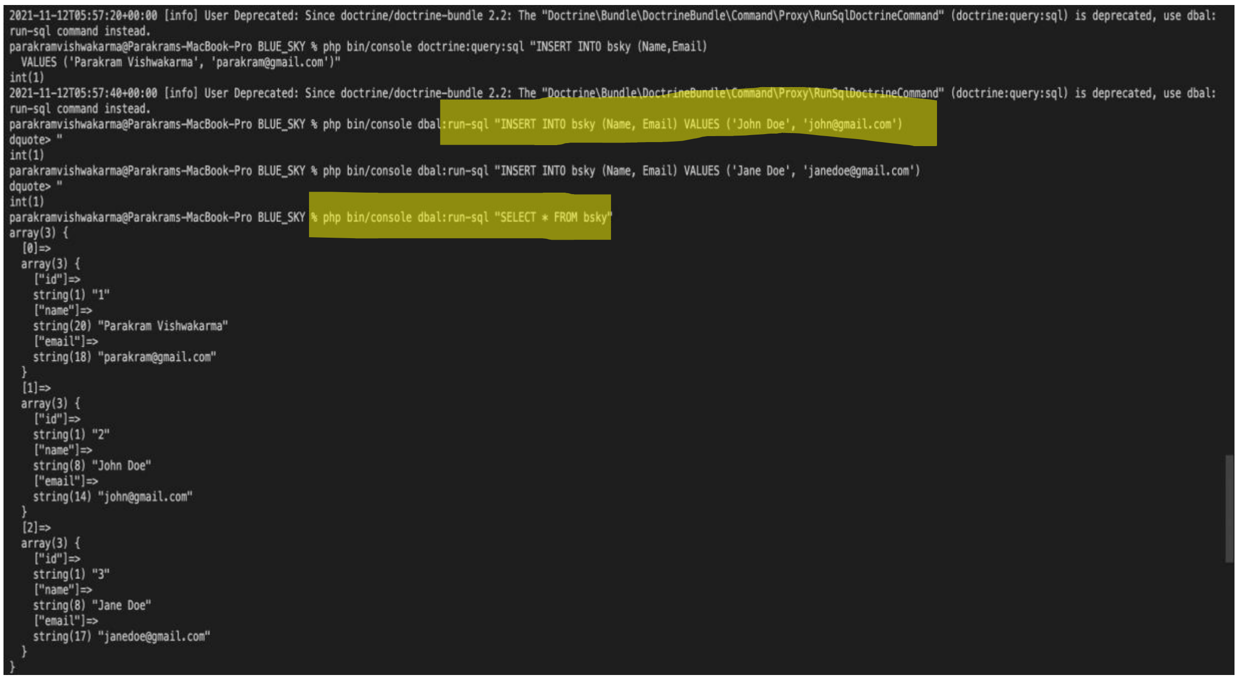


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(30)	utf8mb4_unicode_ci		No	None			Change Drop More
3	email	varchar(30)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More

The default id field is added as visible above and set to auto-increment

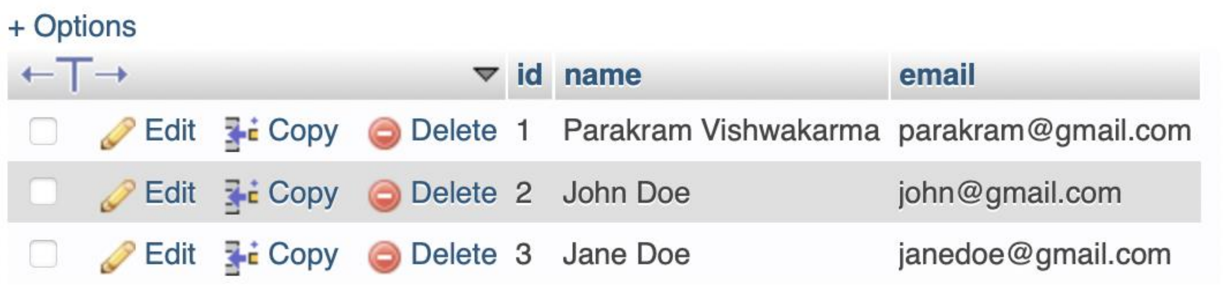
CONSOLE INTERACTION

Now that the DBAL is set with the connection to a database and a table and fields. We can use doctrine to perform SQL queries through the console and interact with the database.



```
2021-11-12T05:57:20+00:00 [info] User Deprecated: Since doctrine/doctrine-bundle 2.2: The "Doctrine\Bundle\DoctrineBundle\Command\Proxy\RunSqlDoctrineCommand" (doctrine:query:sql) is deprecated, use dbal:
run-sql command instead.
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console doctrine:query:sql "INSERT INTO bsky (Name,Email)
  VALUES ('Parakram Vishwakarma', 'parakram@gmail.com')"
int(1)
2021-11-12T05:57:40+00:00 [info] User Deprecated: Since doctrine/doctrine-bundle 2.2: The "Doctrine\Bundle\DoctrineBundle\Command\Proxy\RunSqlDoctrineCommand" (doctrine:query:sql) is deprecated, use dbal:
run-sql command instead.
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console dbal:run-sql "INSERT INTO bsky (Name, Email) VALUES ('John Doe', 'john@gmail.com')
dquote> "
int(1)
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console dbal:run-sql "INSERT INTO bsky (Name, Email) VALUES ('Jane Doe', 'janedoe@gmail.com')
dquote> "
int(1)
parakramvishwakarma@Parakrams-MacBook-Pro BLUE_SKY % php bin/console dbal:run-sql "SELECT * FROM bsky"
array(3) {
  [0]=>
  array(3) {
    ["id"]=>
    string(1) "1"
    ["name"]=>
    string(20) "Parakram Vishwakarma"
    ["email"]=>
    string(18) "parakram@gmail.com"
  }
  [1]=>
  array(3) {
    ["id"]=>
    string(1) "2"
    ["name"]=>
    string(8) "John Doe"
    ["email"]=>
    string(14) "john@gmail.com"
  }
  [2]=>
  array(3) {
    ["id"]=>
    string(1) "3"
    ["name"]=>
    string(8) "Jane Doe"
    ["email"]=>
    string(17) "janedoe@gmail.com"
  }
}
```

Highlighted above are the examples of inserting data and selecting and printing data from the database using the console.



+ Options

				id	name	email
☐	Edit	Copy	Delete	1	Parakram Vishwakarma	parakram@gmail.com
☐	Edit	Copy	Delete	2	John Doe	john@gmail.com
☐	Edit	Copy	Delete	3	Jane Doe	janedoe@gmail.com

This phpMyAdmin used to visualize the database and shows us the insertion has worked from the console.