System obsługi pracowników NFM

Mateusz Król, Arkadiusz Glensk

W-4, PWr

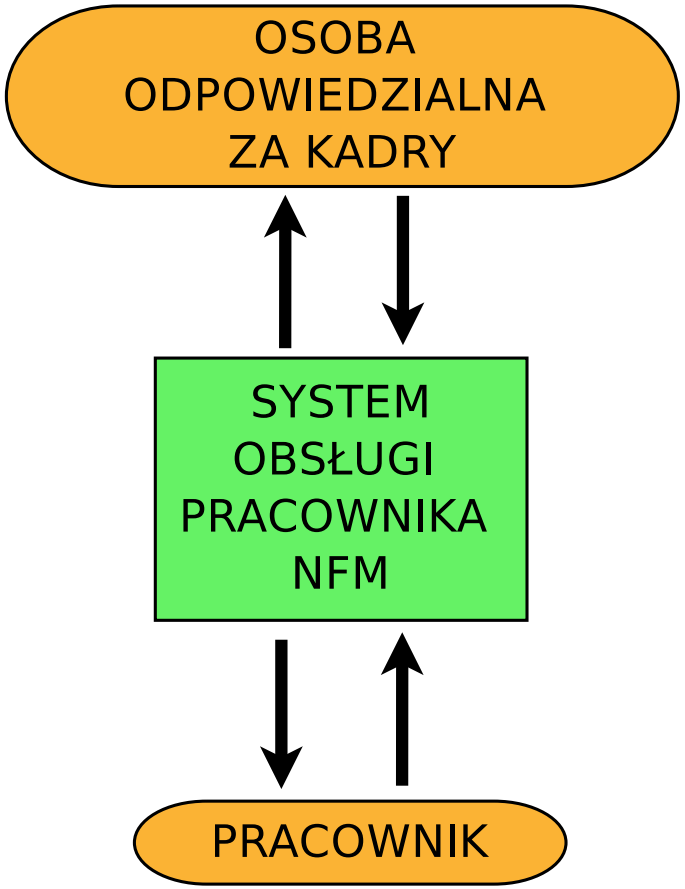
12 czerwca 2018

Bazy danych

Cel stworzenia projektu

Stworzenie znacznie lepszego sposobu rejerstracji i obsługi pracowników NFM, który opiera się na arkuszu kalkulacyjnym zamieszczonym na dysku w chmurze.

Cel stworzenia projektu



Wymagania użytkowe działania systemu

1. Swobodny system rejestracji nowych pracowników
2. Kompleksowa obsługa wydarzeń oraz pracowników
3. Interfejs musi zostać maksymalnie uproszczony ze względu na starszą kadrę pracowniczą

Wymagania funkcjonalne systemu

1. Sprawdzanie obecności pracowników
2. Kontrola miejsc pracy danego wydarzenia
3. Konta pracowników oraz przełożonych muszą różnić się uprawnieniami
4. Cykliczne rozsyłanie maili z informacjami o dostępnych wydarzeniach
5. Rejestracja czasu pracy
6. Delgowanie zadań przez przełożonych (z storny pracowników: wysyłanie żądania o zmianie zadania)

Prace, które się złożyły na wykonanie projektu

- Stworzenie modelu baz danych
- rozwój programuw w zwykłym php
- stworzenie wersji programu opartą o symfony oraz doctrine

Narzędzia wykorzystane przy budowie projektu

- Symfony - framework
- Doctrine - ORM

Symfony

Framework, który oparty jest na wzorcu programistycznym MVC (model-view-controller) w sposób obiektowy. Jest on niezależny od systemu bazodanowego - możemy pracować zarówno na bazach danych SQL jak i NOSQL (jak MongoDB).

Czym jest ORM ?

eng. objected-relational mapping Czyli ?

Czym jest ORM ?

Definicja nieformalna: *Jest to tak jakby most pomiędzy bazą danych a programem. Stworzony poprzez mapowanie rekordów baz dnaych do obiektów w programie.*

Czym jest ORM ?

Definicja formalna: *Sposób odwzorowania obiektowej architektury systemu informatycznego na bazę danych (lub inny element systemu) o relacyjnym charakterze.*

Inne możliwe frameworki do PHP

- Laravel
- CakePHP
- Zend
- Code lgniter

Inne ORMy współpracujące z PHP

- Propel
- PHP Data Mapper
- Simple ORM
- LessQL

Tworzenie bazy danych

Konfiguracja bazy danych

`apliakcja/config/parameters.yml`

Tworzenie bazy danych

`php bin/console doctrine:database:create`

Koniguracja systemu kodowania

Chodzi o ustawienie systemu UTF-8 jako domyślny w pliku konfiguracjynym (nie koniecznie parameters.yml jak poprzednio)

Tworzenie kompletnych encji

Kreowanie encji

```
php bin/console doctrine:generate:entity
```

Kontrola mapowań

```
php bin/console doctrine:schema:validate
```

I już został tylko update do całej bazy danych

```
php bin/console doctrine:schema:update --force
```

Fragment kodu przedstawiający stworzoną encje

Listing 1: Przykładowy kod encji

```php
<?php
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\Table(name="events")
 */
class Post
{

    /**
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     * @ORM\Column(type="integer")
     */
    private $id;


    /**
     * @ORM\Column(type="string")
     */
```

Relacje baz danych

Listing 2: Przykład relacji baz danych (dwukierunkowa)

```php
<?php
/** @Entity */
class Customer
{

    /**
     *
     * @OneToOne(targetEntity="Cart", mappedBy="customer")
     */
    private $cart;

    // ...
}

/** @Entity */
class Cart
{

    /**
     *
     * @OneToOne(targetEntity="Customer", inversedBy="cart")
     * @JoinColumn(name="customer_id", referencedColumnName="id")
     */
    private $customer;

    // ...
}
```

Dodawanie obiektów

```
1 temp = this->getDoctrine()->getManager();
2 user = new User();
3 user->setName('Bogumił');
4 temp->persist(user);
5 temp->flush();
```

Dodawanie obiektów

```
1 temp = this->getDoctrine()->getManager();
2 user = new User();
3 user->setName('Bogumił');
4 temp->persist(user);
5 temp->flush();
```

Dodawanie obiektów

```
1 temp = this->getDoctrine()->getManager();
2 user = new User();
3 user->setName('Bogumił');
4 temp->persist(user);
5 temp->flush();
```

Dodawanie obiektów

```
1 temp = this->getDoctrine()->getManager();
2 user = new User();
3 user->setName('Bogumił');
4 temp->persist(user);
5 temp->flush();
```

Dodawanie obiektów

```
1 temp = this->getDoctrine()->getManager();
2 user = new User();
3 user->setName('Bogumił');
4 temp->persist(user);
5 temp->flush();
```

Dodawanie obiektów

1. `temp = this->getDoctrine()->getManager();`
2. `user = new User();`
3. `user->setName('Bogumił');`
4. `temp->persist(user);`
5. `temp->flush();`

Pobieranie elementu

1. `temp = this->getDoctrine()->getRepository(User::class)`
2. `->find(userId);`

Zmiana elementu rekordu polega na:
`wyszukanie -> zmianie za pomocą setera -> flush()`

Pobieranie elementu

1. `temp = this->getDoctrine()->getRepository(User::class)`
2. `->find(userId);`

Zmiana elementu rekordu polega na:
`wyszukanie -> zmianie za pomocą setera -> flush()`

Pobieranie elementu

1. `temp = this->getDoctrine()->getRepository(User::class)`
2. `->find(userId);`

Zmiana elementu rekordu polega na:
`wyszukanie -> zmianie za pomocą setera -> flush()`

Kwerendy

Dwie możliwości:

1. DQL
   Pisanie zapytań, składnia bardzo podobna do SQL
2. Doctrine Query Bulider
   Zapytanie jest kreowane na podstawie funkcji jakich używiliśmy w Buliderze

Wywołanie kwerendy polega na uruchomienie jej poprzez funkcję `getResult()`

Kwerendy

Dwie możliwości:

1. DQL
   Pisanie zapytań, składnia bardzo podobna do SQL
2. Doctrine Query Bulider
   Zapytanie jest kreowane na podstawie funkcji jakich używiliśmy w Buliderze

Wywołanie kwerendy polega na uruchomienie jej poprzez funkcję `getResult()`

Kwerendy

Dwie możliwości:

1. DQL
   Pisanie zapytań, składnia bardzo podobna do SQL
2. Doctrine Query Bulider
   Zapytanie jest kreowane na podstawie funkcji jakich używiliśmy w Buliderze

Wywołanie kwerendy polega na uruchomienie jej poprzez funkcję `getResult()`

Kwerendy

Dwie możliwości:

1. DQL
   Pisanie zapytań, składnia bardzo podobna do SQL
2. Doctrine Query Bulider
   Zapytanie jest kreowane na podstawie funkcji jakich używiliśmy w Buliderze

Wywołanie kwerendy polega na uruchomienie jej poprzez funkcję `getResult()`

Kwerendy

Dwie możliwości:

1. DQL
   Pisanie zapytań, składnia bardzo podobna do SQL
2. Doctrine Query Bulider
   Zapytanie jest kreowane na podstawie funkcji jakich używiliśmy w Buliderze

Wywołanie kwerendy polega na uruchomienie jej poprzez funkcję `getResult()`

Dziękujemy za uwagę.