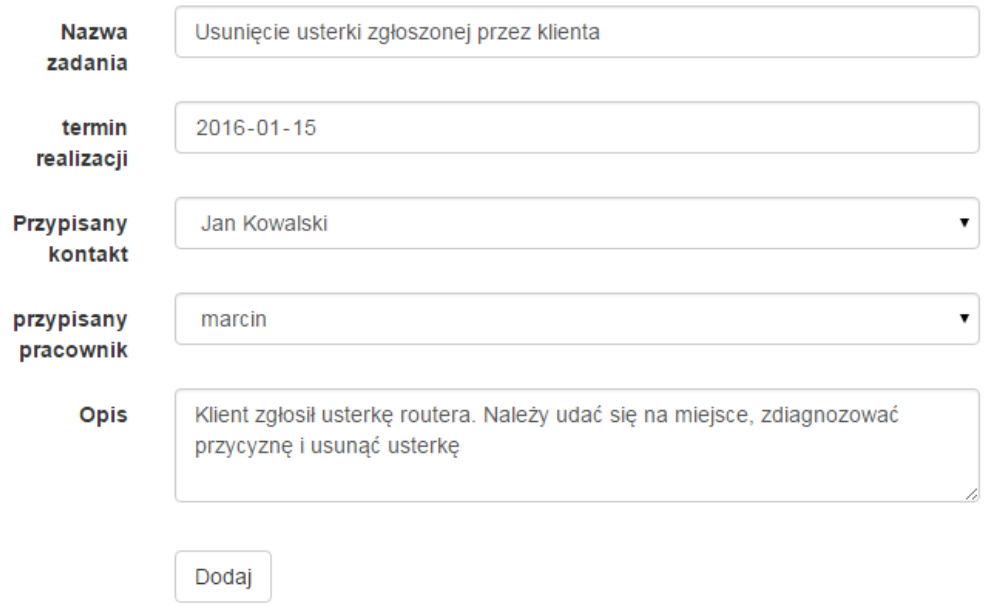
Utrwalanie danych w bazie danych przy użyciu technologii MongoDB, Express.js Angular.js oraz Node.js.

Aplikacja została zbudowana w oparciu o MEAN stack czyli stosu technologii Java Script służących do budowania aplikacji internetowych.

Komponenty MEAN stack:

- **M**ongoDB,- baza danych NoSQL
- **E**xpress.js – framework, który zapewnia łatwość budowania serwera, uruchamiany w środowisku Node.js
- **A**ngular.js –Framework JavaScript MVC, który działa w przeglądarce
- **N**ode.js – środowisko uruchomieniowe dla aplikacji serwerowych sterowanych zdarzeniami.

Przewagą takiego podejścia jest to, że zarówno po stronie serwerowej jak i klienta zostaje użyty jeden język programowania.



Rysunek 1. Zrzut ekranu przedstawiający formularz dodawania nowego zadania.

Rysunek 1 przedstawia formularz dodawania nowego zadania. Należy nadać nazwę, termin realizacji, przypisany kontakt, którego dotyczyć będzie zadanie, należy też wskazać pracownika oraz ewentualnie dodać opis zadania.

```
<form class="form-horizontal" ng-submit="addTask()">
    <div class="form-group">
        <label for="inputTaskName" class="col-sm-2 control-label">Nazwa
zadania</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" id="inputTaskName"
placeholder="Nazwa zadania" ng-model="taskName"></div>
    <div class="form-group">
        <div class="col-sm-offset-2 col-sm-10">
            <button type="submit" class="btn btn-default">Dodaj</button>
        </div>
    </div>
</form>
```

Listing 1. Fragment kodu formularza służącego do dodawania nowego zadania (plik task.html).

Listing 1 przedstawia fragment kodu służącego do obsługi formularza. W formularzu znajdują się dyrektywy rozpoznawane przez Angular.js Dzięki konstrukcji: `ng-submit="addTask()` w momencie wciśnięcia przycisku **Dodaj** zostanie uruchomiona funkcja o nazwie addTask, znajduje się ona w kontrolerze o nazwie **addTaskCtrl**. Cechą charakterystyczną dla Angulara jest dwustronne bindowanie zmiennych, tak więc wpisując tekst w pole **Nazwa zadania:** „Usunięcie usterki zgłoszonej przez klienta" poprzez dyrektywę `ng-model="taskName"` będzie on niezwłocznie widoczny w zmiennej $scope.taskName w kontrolerze.

```
app.controller('addTaskCtrl', function($scope,$http){

$scope.addTask= function(){
    $http.post('http://localhost:3000/task',
    {"name":$scope.taskName,"time_limit":$scope.timeLimit,
"desc":$scope.taskDesc, "assigned_contact":$scope.assignedContact,
"assigned_employee":$scope.assignedEmpl})
.success(function(data){
          console.log(data);
        });
    };
});
```

Listing 2. Kod kontrolera odpowiadającego za dodawanie zadania(plik app.js).

Usługa $http wykonuje żądania Ajax. Metoda post używana jest do przesyłania danych do serwera celem zapisania ich w bazie danych. Jako argumenty usługa przyjmuje adres URL oraz dane w postaci JSON. Success odpowiada za akcję wykonywaną gdy serwer zwróci status iż wszystko się powiodło.

W pliku app.js znajdującym się w głównym folderze aplikacji należy dodać adres bazy danych, tak aby Moongose mogło się z nią połączyć:

```
mongoose.connect('mongodb://localhost/minicrm');
```

```
var express = require('express');
var router = express.Router();
require('../models/Tasks');
var Task= mongoose.model('Task');
router.post('/task', function(req, res) {
    var task = new Task(req.body);
    console.log(req.body);
    task.save(function(err, contact){
        if(err){ return next(err); }
        res.json({status:"ADDED"});
    });
});
router.put('/task',function(req, res){

    Task.findOne({ _id: req.body._id }, function (err, task){
        task.status= req.body.status;
        if(typeof req.body.comment!='undefined')
{task.comments.push({comment: req.body.comment,user:req.body.user});}
        task.save();
    });
    return res.status(200).json({message: 'OK'});
});
```

Listing 3. API zbudowane w oparciu o express.js(plik api.js)

Po wysłaniu przez aplikację internetową danych trafiają one do serwera, żądanie jest identyfikowane po adresie URL i obsługa przekazywana jest do odpowiedniej funkcji. Na listingu 3 przedstawiony jest

fragment pliku api.js. Najpierw należy zaimportować moduł express.js, do tego celu służy konstrukcja: `var express = require('express')` Następnie tworzony jest router. Konieczne jest zaimportowanie modelu czyli schematu zadania zapisywanego do bazy danych MongoDB. deklarując nowy element API należy podać jakiej metody protokołu http będziemy używać. W tym przypadku jest to metoda POST. Tworzone jest nowe zadanie, dane, które zostały wysłane znajdują się w zmiennej req.**body.** Na obiekcie task wywołana jest metoda save. To właśnie w tym miejscu następuje zapis danych do bazy MongoDB. Odbywa się on za pomocą narzędzia Moongose. Metoda PUT protokołu http służy do aktualizacji zawartości bazy danych. Najpierw przy użyciu dostarczonego ID konkretnego zadania następuje odszukanie go w bazie danych: `Task.findOne({ _id: req.body._id}, function (err, task)` Następnie do funkcji przekazywane jest znalezione zadanie, dane w poszczególnych zmiennych zostają zaktualizowane, a następnie poprzez `task.save()` zaktualizowane dane są zapisane w bazie.

```
var mongoose = require('mongoose');

var TaskSchema = new mongoose.Schema({
    name :{type:  String, required: true},
    time_limit:{type: Date},
    created_at: { type: Date,
        default: Date.now},
    desc: String,
    assigned_contact:  String,
    assigned_employee:  String,
    comments: [
        { created_at: {type: Date, default: Date.now},
            user: String,
            comment: String
        }
    ],
    status: { type: String,
        default: 'Nowe'
    }
});
mongoose.model('Task', TaskSchema);
```

Listing 4. Schemat zadania(plik Tasks.js)

Listing 4 przedstawia schemat wykorzystywany do zapisu zadań do bazy danych. Używając tego mechanizmu mamy pewność, że zapisywane dane będą miały spójną strukturę. Jest to bardzo ważne ponieważ nierelacyjne bazy danych nie mają z góry ściśle określonej struktury przechowywanych danych.