Rest API

Code

Code is separated to a couple of segments. First is reserved to definition of used bundles in composer.json and AppKernel.php. In those file there are definitions for FOSRestBundle, FOSAuthServerBundle and JMSSerializerBundle.

composer.json

```
...
"require": {
    "...
    "friendsofsymfony/rest-bundle": "^1.8",
    "friendsofsymfony/oauth-server-bundle": "^1.5",
    "jms/serializer-bundle": "^1.1"
},
...
```

AppKernel.php

```
$bundles = array(
    ...
    new JMS\SerializerBundle\JMSSerializerBundle(),
    new FOS\RestBundle\FOSRestBundle(),
    new FOS\OAuthServerBundle\FOSOAuthServerBundle(),

    ...
);
```

Additionally, there are configuration for REST and OAuth2 bundles, defined in app/config/commons/all/config.yml. Classes created inside folder Mealz\RestBundle\Entity are created as desired by FOSAuthServerBundle documentation, and they are used to work with tables in database, so it can uses different clients to login, and to retrieve access tokens and refresh tokens.

config.yml

```
fos_rest:
    param_fetcher_listener: true
    body_listener: true
    format_listener: true
    view:
        view_response_listener: 'force'
        exception_wrapper_handler: Mealz\RestBundle\Handler\ExceptionWrapperHandler
        formats:
            xml: true
            json : true
        templating_formats:
            html: true
        force_redirects:
            html: true
        failed_validation: HTTP_BAD_REQUEST
        default_engine: twig
    exception:
        enabled: true
    routing_loader:
        default_format: json

fos_oauth_server:
    db_driver:           orm
    client_class:        Mealz\RestBundle\Entity\Client
    access_token_class:  Mealz\RestBundle\Entity\AccessToken
    refresh_token_class: Mealz\RestBundle\Entity\RefreshToken
    auth_code_class:     Mealz\RestBundle\Entity\AuthCode
    service:
        user_provider: mealz_user.provider.login
```

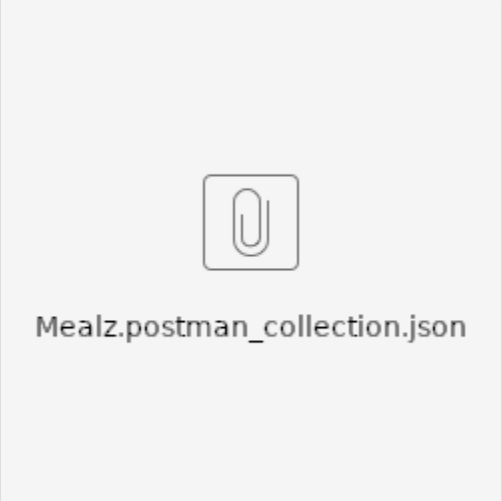
Complete PHP code is placed inside separate bundle (RestBundle), inside folder src/Mealz/RestBundle, and it's related only to code used inside Controller section (BaseController, MealController, ParticipationController and WeekController). Routing for REST is defined inside src/Mealz/RestBundle/Resources /config/routing.yml.

routing.yml

```
mealz_rest_week_current:
    type: rest
    path:     /week/active
    defaults: { _controller: MealzRestBundle:Week:active }

mealz_rest_participant_delete:
    type: rest
    path:     /participant/{participantId}
    defaults: { _controller: MealzRestBundle:Participant:delete }
    methods:  [DELETE]

mealz_rest_participant_confirm:
    type: rest
    path:     /participant/{participantId}
    defaults: { _controller: MealzRestBundle:Participant:confirm }
    methods:  [PUT]

mealz_rest_participant_today:
    type: rest
    path:     /participant/today
    defaults: { _controller: MealzRestBundle:Participant:today }

mealz_rest_participant_add:
    type: rest
    path:     /participant/{date}/{dishId}
    defaults: { _controller: MealzRestBundle:Meal:add }
    methods:  [POST]
```

How to use

To check REST functionality, it's strongly recommended to use application called Postman. Bellow there is configuration file with all defined requests for REST API that should be imported via Postman.



After import, first request that should be called is login, which will return access token that should be used in other calls. To change user credentials in login process, there are parameters in body section of request, so different users can be used to login on system. Before request is sent, client should be generated in database. For test purpose, dummy client can be inserted into database, by using next query:

```
INSERT INTO oauth2_clients VALUES (NULL, '3bcbxd9e24g0gk4swg0kwgcwg4o8k8g4g888kwc44gcc0gwwk4', 'a:0:{}',
'4ok2x70rlfokc8g0wws8c8kwcokw80k44sg48goc0ok4w0so0k', 'a:1:{i:0;s:8:"password";}');
```

After login process is finished, resulting access token should be placed into each request (Headers section), so it can be used by endpoint to fetch user data.

All possible REST requests:

1. Login: /oauth/v2/token, POST - simply provide user credentials and make sure that dummy client is created
2. Active week: /rest/v1/week/active, GET - just update access token and information about meals for current and next week will be provided (together with information about participation for logged in user)
3. Active participation: /rest/v1/participant/today, GET - just update access token and information about participation for current day will be provided
4. Add participation: /rest/v1/participant/{date}/{dishId}, POST - update access token and provide date and dish ID (it can be observed via database), and new participation will be created and information about it will be provided
5. Delete participation: /participant/{participantId}, DELETE - update access token and provide participation ID (it can be observed via database), and participation will be deleted and information about it will be provided