Code source - API REST

Jonathan Borel-Jaquet

10 juin 2021

Listings

1 ./Sources/app/Controllers/AbsenceController.php . . . 3
2 ./Sources/app/Controllers/AppointmentController.php . . . 7
3 ./Sources/app/Controllers/DocumentController.php . . . 13
4 ./Sources/app/Controllers/DogController.php . . . 20
5 ./Sources/app/Controllers/HelperController.php . . . 25
6 ./Sources/app/Controllers/ResponseController.php . . . 34
7 ./Sources/app/Controllers/ScheduleOverrideController.php . . . 41
8 ./Sources/app/Controllers/TimeSlotController.php . . . 44
9 ./Sources/app/Controllers/UserController.php . . . 48
10 ./Sources/app/Controllers/WeeklyScheduleController.php . . . 56
11 ./Sources/app/DataAccessObject/DAOAbsence.php . . . 60
12 ./Sources/app/DataAccessObject/DAOAppointment.php . . . 63
13 ./Sources/app/DataAccessObject/DAODocument.php . . . 69
14 ./Sources/app/DataAccessObject/DAODog.php . . . 75
15 ./Sources/app/DataAccessObject/DAOScheduleOverride.php . . . 79
16 ./Sources/app/DataAccessObject/DAOTimeSlot.php . . . 83
17 ./Sources/app/DataAccessObject/DAOUser.php . . . 93
18 ./Sources/app/DataAccessObject/DAOWeeklySchedule.php . . . 99
19 ./Sources/app/Models/Absence.php . . . 104
20 ./Sources/app/Models/Appointment.php . . . 105
21 ./Sources/app/Models/Document.php . . . 105
22 ./Sources/app/Models/Dog.php . . . 106
23 ./Sources/app/Models/ScheduleOverride.php . . . 107
24 ./Sources/app/Models/TimeSlot.php . . . 108
25 ./Sources/app/Models/User.php . . . 108
26 ./Sources/app/Models/WeeklySchedule.php . . . 109
27 ./Sources/app/System/Constants.php . . . 110
28 ./Sources/app/System/DatabaseConnector.php . . . 110
29 ./Sources/bootstrap.php . . . 111
30 ./Sources/public/v1/absences/index.php . . . 112
31 ./Sources/public/v1/appointments/downloadNoteGraphical/index.php . . . 113
32 ./Sources/public/v1/appointments/index.php . . . 114
33 ./Sources/public/v1/appointments/uploadNoteGraphical/index.php . . . 116
34 ./Sources/public/v1/connection/index.php . . . 116
35 ./Sources/public/v1/documents/downloadDocument/index.php . . . 117
36 ./Sources/public/v1/documents/index.php . . . 118
37 ./Sources/public/v1/dogs/downloadPicture/index.php . . . 120
38 ./Sources/public/v1/dogs/index.php . . . 120
39 ./Sources/public/v1/dogs/uploadPicture/index.php . . . 122
40 ./Sources/public/v1/index.php . . . 123
41 ./Sources/public/v1/plannings/index.php . . . 123
42 ./Sources/public/v1/scheduleOverrides/index.php . . . 124
43 ./Sources/public/v1/timeSlots/index.php . . . 125
44 ./Sources/public/v1/users/educators/index.php . . . 127
45 ./Sources/public/v1/users/index.php . . . 127
46 ./Sources/public/v1/users/me/changePassword/index.php . . . 129
47 ./Sources/public/v1/users/me/index.php . . . 130
48 ./Sources/public/v1/weeklySchedules/index.php . . . 131
49 ./Sources/resources/template/conditions//textunderscoreregistration.php . . . 132
50 ./Sources/resources/template/iCalendar//textunderscoreappointment.php . . . 153

Listing 1 – ./Sources/app/Controllers/AbsenceController.php

```php
<?php
/**
 * AbsenceController.php
 *
 * Controller of the Absence model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAOAbsence;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\Models\Absence;
use App\System\Constants;

class AbsenceController {

    private DAOAbsence $DAOAbsence;
    private DAOUser $DAOUser;

    /**
     *
     * Constructor of the AbsenceController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAOAbsence = new DAOAbsence($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all absences in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllAbsences()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allAbsences = $this->DAOAbsence->findAll(false,$userAuth->id);

        return ResponseController::successfulRequest($allAbsences);
    }

    /**
     *
     * Method to return a absence in JSON format.
     *
     * @param int  $id The absence identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getAbsence(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $absence = $this->DAOAbsence->find($id,$userAuth->id);

        if (is_null($absence)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($absence);
    }

    /**
     *
     * Method to create a absence.
     *
     * @param Absence $absence The absence model object
     * @return string The status and the body in JSON format of the response
     */
    public function createAbsence(Absence $absence)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $absence->id_educator = $userAuth->id;

        if (!$this->validateAbsence($absence)) {
```

```php
            return ResponseController::unprocessableEntityResponse();
        }

        if (!HelperController::validateDateFormat($absence->date_absence_from) || !
            HelperController::validateDateFormat($absence->date_absence_to) ) {
            return ResponseController::invalidDateFormat();
        }

        if (!HelperController::validateChornologicalTime($absence->
            date_absence_from,$absence->date_absence_to)) {
            return ResponseController::chronologicalDateProblem();
        }

        $this->DAOAbsence->insert($absence);

        return ResponseController::successfulCreatedRessource();
    }

    /**
     *
     * Method to update a absence.
     *
     * @param Absence $absence The absence model object
     * @return string The status and the body in JSON format of the response
     */
    public function updateAbsence(Absence $absence)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $actualAbsence = $this->DAOAbsence->find($absence->id,$userAuth->id);

        if (is_null($actualAbsence)) {
            return ResponseController::notFoundResponse();
        }

        $actualAbsence->date_absence_from = $absence->date_absence_from ??
            $actualAbsence->date_absence_from;
        $actualAbsence->date_absence_to = $absence->date_absence_to ??
            $actualAbsence->date_absence_to;
        $actualAbsence->description = $absence->description ?? $actualAbsence->
            description;

        if (!HelperController::validateDateFormat($actualAbsence->date_absence_from
            ) || !HelperController::validateDateFormat($actualAbsence->
            date_absence_to) ) {
            return ResponseController::invalidDateFormat();
        }

        if (!HelperController::validateChornologicalTime($actualAbsence->
            date_absence_from,$actualAbsence->date_absence_to)) {
            return ResponseController::chronologicalDateProblem();
        }

        $this->DAOAbsence->update($actualAbsence);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to delete a absence.
     *
     * @param int  $id The absence identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteAbsence($id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $absence = $this->DAOAbsence->find($id,$userAuth->id);

        if (is_null($absence)) {
            return ResponseController::notFoundResponse();
        }

        $this->DAOAbsence->delete($absence);

        return ResponseController::successfulRequest(null);
    }

     /**
     *
     * Method to check if the absence required fields have been defined for the
         creation.
     *
     * @param Absence $absence The absence model object
     * @return bool
     */
    private function validateAbsence(Absence $absence)
    {
        if ($absence->date_absence_from == null) {
            return false;
        }

        if ($absence->date_absence_to == null) {
            return false;
```

```php
        }

        if ($absence->id_educator == null) {
            return false;
        }

        return true;
    }
}
```

Listing 2 – ./Sources/app/Controllers/AppointmentController.php

```php
<?php
/**
 * AppointmentController.php
 *
 * Controller of the Appointment model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAOAppointment;
use App\DataAccessObject\DAOTimeSlot;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\Models\Appointment;
use App\System\Constants;

class AppointmentController {

    private DAOAppointment $DAOAppointment;
    private DAOTimeSlot $DAOTimeSlot;
    private DAOUser $DAOUser;


    /**
     *
     * Constructor of the AppointmentController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAOAppointment = new DAOAppointment($db);
        $this->DAOTimeSlot = new DAOTimeSlot($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all Appointments in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllAppointments()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth)) {
            return ResponseController::unauthorizedUser();
        }

        if($userAuth->code_role == Constants::ADMIN_CODE_ROLE){
            $allAppointments = $this->DAOAppointment->findAll(null,$userAuth->id);
        }
        else{
            $allAppointments = $this->DAOAppointment->findByUserId($userAuth->id);
        }

        return ResponseController::successfulRequest($allAppointments);
    }

    /**
     *
     * Method to return a appointment in JSON format.
     *
     * @param int $id The appointment identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getAppointment(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $appointment = $this->DAOAppointment->find($id);

        if (is_null($appointment)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($appointment);
    }

    /**
     *
     * Method to create a appointment.
     *
     * @param Appointment $appointment The appointment model object
     * @return string The status and the body in JSON format of the response
     */
    public function createAppointment(Appointment $appointment)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth)) {
            return ResponseController::unauthorizedUser();
        }

        if (!$this->validateAppointment($appointment)) {
            return ResponseController::unprocessableEntityResponse();
        }

        $userCustomer = $this->DAOUser->find($appointment->user_id_customer);
        $userEducator = $this->DAOUser->find($appointment->user_id_educator);

        if (is_null($userCustomer) || is_null($userEducator) || $userCustomer->
            code_role != Constants::USER_CODE_ROLE || $userEducator->code_role !=
            Constants::ADMIN_CODE_ROLE) {
            return ResponseController::notFoundResponse();
        }

        if (!HelperController::validateDateTimeFormat($appointment->
            datetime_appointment)) {
            return ResponseController::invalidDateTimeFormat();
        }

        $datetime_start = new \DateTime($appointment->datetime_appointment, new \
            DateTimeZone("Europe/Berlin"));
        $datetime_end = clone $datetime_start;
        $datetime_end->modify('+'.$appointment->duration_in_hour." hours");

        $date = $datetime_start->format("Y-m-d");
        $time_start = $datetime_start->format("H:i:s");
        $time_end = $datetime_end->format("H:i:s");

        if ($userAuth->code_role == Constants::USER_CODE_ROLE) {

            if (!$this->DAOTimeSlot->findAppointmentSlotsForEducator($date,
                $time_start,$time_end,$appointment->user_id_educator)) {
                return ResponseController::invalidAppointment();
            }

        }

        $this->DAOAppointment->insert($appointment);

        $filename =  "iCal-" . $datetime_start->format("Ymd"). ".ics";

        $educator_fullname = $userEducator->firstname . " " . $userEducator->
            lastname;

        HelperController::sendMailWithICSFile($datetime_start, $datetime_end,
            $educator_fullname, $userCustomer->email, $filename);

        return ResponseController::successfulCreatedRessource();
    }

    /**
     *
     * Method to update a appointment.
     *
     * @param int  $id The appointment identifier
     * @return string The status and the body in JSON format of the response
     */
    public function updateAppointment(Appointment $appointment)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $actualAppointment = $this->DAOAppointment->find($appointment->id);

        if (is_null($actualAppointment)) {
            return ResponseController::notFoundResponse();
        }

        $actualAppointment->note_text = $appointment->note_text ??
            $actualAppointment->note_text;
        $actualAppointment->summary = $appointment->summary ?? $actualAppointment->
            summary;

        $this->DAOAppointment->update($actualAppointment);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to delete a appointment.
     *
     * @param int  $id The appointment identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteAppointment(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth)) {
            return ResponseController::unauthorizedUser();
        }

        $appointment = $this->DAOAppointment->find($id);

        if (is_null($appointment)) {
            return ResponseController::notFoundResponse();
        }

        if ($userAuth->id == $appointment->user_id_educator xor $userAuth->id !=
            $appointment->user_id_customer ) {
            return ResponseController::unauthorizedUser();
        }

        $this->DAOAppointment->delete($appointment->id,$userAuth->id);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to upload a graphical note.
     *
     * @return string The status and the body in JSON format of the response
     */
    public function uploadNoteGraphical()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        if (!isset($_FILES["note_graphical"]) || !is_uploaded_file($_FILES["
            note_graphical"]["tmp_name"]) || !isset($_POST["appointment_id"])) {
            return ResponseController::unprocessableEntityResponse();
        }

        $appointment = $this->DAOAppointment->find($_POST["appointment_id"]);

        if (is_null($appointment)) {
            return ResponseController::notFoundResponse();
        }

        switch ($_FILES["note_graphical"]["type"]) {
            case Constants::IMAGE_TYPE_PNG:
                HelperController::pngTojpegConverter($_FILES["note_graphical"]["
                    tmp_name"]);
                break;
            case Constants::IMAGE_TYPE_JPEG:
```

```php
                break;
            default:
                return ResponseController::imageFileFormatProblem();
                break;
        }

        $tmp_file = $_FILES["note_graphical"]["tmp_name"];
        $img_name = HelperController::generateRandomString();
        $upload_dir = HelperController::getDefaultDirectory()."storage/app/
            graphical_note/".$img_name.".jpeg";

        if (!is_null($appointment->note_graphical_serial_id)) {
            $filename = HelperController::getDefaultDirectory()."storage/app/
                graphical_note/".$appointment->note_graphical_serial_id.".jpeg";
            if (file_exists($filename)) {
                unlink($filename);
            }
        }

        if (!move_uploaded_file($tmp_file,$upload_dir)) {
            return ResponseController::uploadFailed();
        }

        $appointment->note_graphical_serial_id = $img_name;

        $this->DAOAppointment->update($appointment);

        return ResponseController::successfulRequest();
    }

    /**
     *
     * Method to download a graphical note.
     *
     * @param string  $serial_id The serial_id of the graphical note
     * @return string The status and the body in JSON format of the response
     */
    public function downloadNoteGraphical(string $serial_id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        if(is_null($this->DAOAppointment->findBySerialId($serial_id))){
            return ResponseController::notFoundResponse();
        }

        $image = HelperController::getDefaultDirectory()."storage/app/
            graphical_note/".$serial_id.".jpeg";

        $imageInfo = getimagesize($image);

        header("Content-Type: " . $imageInfo["mime"]);
        header("Content-Length: " . filesize($image));
        readfile($image);

        return ResponseController::successfulRequest();
    }

     /**
     *
     * Method to check if the appointment required fields have been defined for the
       creation.
     *
     * @param Appointment $appointment The appointment model object
     * @return bool
     */
    private function validateAppointment(Appointment $appointment)
    {
        if ($appointment->datetime_appointment == null) {
            return false;
        }

        if ($appointment->duration_in_hour == null) {
            return false;
        }

        if ($appointment->user_id_customer == null) {
            return false;
        }

        if ($appointment->user_id_educator == null) {
            return false;
        }

        return true;
    }
}
```

Listing 3 – ./Sources/app/Controllers/DocumentController.php

```php
<?php
/**
 * DocumentController.php
 *
 * Controller of the Document model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAODocument;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\Models\Document;
use App\System\Constants;

```

```php
class DocumentController {

    private DAODocument $DAODocument;
    private DAOUser $DAOUser;

    /**
     *
     * Constructor of the DocumentController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAODocument = new DAODocument($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all documents in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllDocuments()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allDocuments = $this->DAODocument->findAll();

        return ResponseController::successfulRequest($allDocuments);
    }

    /**
     *
     * Method to return a document in JSON format.
     *
     * @param int $id The document identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getDocument(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $document = $this->DAODocument->find($id);

        if (is_null($document)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($document);
    }

    /**
     *
     * Method to create a document.
     *
     * @param Document $document The document model object
     * @return string The status and the body in JSON format of the response
     */
    public function createDocument(Document $document)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        if (!$this->validateDocument($document)) {
            return ResponseController::unprocessableEntityResponse();
        }

        if (!HelperController::validateDocumentTypeFormat($document->type)) {
            return ResponseController::invalidDocumentTypeFormat();
        }

        $user = $this->DAOUser->find($document->user_id);

        if (is_null($user)) {
            return ResponseController::notFoundResponse();
        }

        $filename = HelperController::generateRandomString();

        switch ($document->type) {
            case Constants::DOCUMENT_TYPE_CONDTIONS_OF_REGISTRATION:

                if (!$this->validateDocumentConditionsRegistration($document)) {
                    return ResponseController::unprocessableEntityResponse();
                }

                if (!HelperController::validatePackageNumber($document->
                    package_number)) {
                    return ResponseController::packageNumberFormatProblem();
                }

                $package_number = $document->package_number;
                $date = date('d/m/Y');
                $document->document_serial_id = $filename;
                $signature_base64 = $document->signature_base64;
                $userfirstname = $user->firstname;
                $userlastname = $user->lastname;

                HelperController::storeConditionsRegistration($filename,
                    $package_number,$date,$signature_base64,$userfirstname,
                    $userlastname);

                $upload_file = HelperController::getDefaultDirectory()."storage/app
                    /conditions_registration/".$filename.".pdf";

                break;

            case Constants::DOCUMENT_TYPE_POSTER:

                if (!isset($_FILES["document"])) {
                    return ResponseController::unprocessableEntityResponse();
                }

                if ($_FILES["document"]["type"] != Constants::TYPE_DOCUMENT_PDF) {
                    return ResponseController::documentTypeNotPdfProblem();
                }

                $tmp_file = $_FILES["document"]["tmp_name"];
                $upload_file = HelperController::getDefaultDirectory()."storage/app
                    /pdf/".$filename.".pdf";

                if (!move_uploaded_file($tmp_file,$upload_file)) {
                    return ResponseController::uploadFailed();
                }

                break;
            default:
                return ResponseController::invalidDocumentTypeFormat();
                break;
        }

        $document->document_serial_id = $filename;

        $this->DAODocument->insert($document);

        HelperController::sendMail("Bonjour et merci de faire confiance à la sociét
            é Douceur de Chien, vous trouverez ci-joint le document qui a été ajout
            é à votre compte, vous pouvez égalament accéder à ce document depuis
            votre compte.","Un nouveau document a été ajouté à votre compte",$user
            ->email,null,$upload_file);

        return ResponseController::successfulCreatedRessource();
    }

```

```php
    /**
     *
     * Method to update a document.
     *
     * @param Document $document The document model object
     * @return string The status and the body in JSON format of the response
     */
    public function updateDocument(Document $document)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (!$userAuth ||  $userAuth->code_role != Constants::ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $actualDocument = $this->DAODocument->find($document->id);

        if (is_null($actualDocument)) {
            return ResponseController::notFoundResponse();
        }

        $actualDocument->document_serial_id = $document->document_serial_id ??
            $actualDocument->document_serial_id;
        $actualDocument->type = $document->type ?? $actualDocument->type;

        if (!HelperController::validateDocumentTypeFormat($actualDocument->type)) {
            return ResponseController::invalidDocumentTypeFormat();
        }

        $this->DAODocument->update($actualDocument);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to delete a document.
     *
     * @param int  $id The document identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteDocument(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (!$userAuth ||  $userAuth->code_role != Constants::ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $document = $this->DAODocument->find($id);

        if (is_null($document)) {
            return ResponseController::notFoundResponse();
        }

        switch ($document->type) {
            case Constants::DOCUMENT_TYPE_CONDTIONS_OF_REGISTRATION:
                $filename = HelperController::getDefaultDirectory()."storage/app/
                    conditions_registration/".$document->document_serial_id.".pdf";

                if (file_exists($filename)) {
                    unlink($filename);
                }
                break;

            case Constants::DOCUMENT_TYPE_POSTER:

                $filename = HelperController::getDefaultDirectory()."storage/app/
                    pdf/".$document->document_serial_id.".pdf";

                if (file_exists($filename)) {
                    unlink($filename);
                }
                break;

            default:
                break;
        }

        $this->DAODocument->delete($document);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to download a document.
     *
     * @param int  $serial_id The document identifier
     * @return string The status and the body in JSON format of the response
     */
    public function downloadDocument(string $serial_id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth)) {
            return ResponseController::notFoundResponse();
        }

        if ($userAuth ||  $userAuth->code_role == Constants::ADMIN_CODE_ROLE) {
            $document = $this->DAODocument->findBySerialId($serial_id);
        }
        else{
            $document = $this->DAODocument->findByUserIdAndSerialId($userAuth->id,
                $serial_id);
        }

        if (is_null($document)) {
            return ResponseController::notFoundResponse();
        }

        header("Content-Type: application/pdf");

        switch ($document->type) {
            case Constants::DOCUMENT_TYPE_CONDTIONS_OF_REGISTRATION:

                $document_data = file_get_contents(HelperController::
                    getDefaultDirectory()."storage/app/conditions_registration/".
                    $serial_id.".pdf");
                break;

            case Constants::DOCUMENT_TYPE_POSTER:

                $document_data = file_get_contents(HelperController::
                    getDefaultDirectory()."storage/app/pdf/".$serial_id.".pdf");
                break;

            default:
                break;
        }

        return ResponseController::successfulRequestWithoutJson($document_data);
    }

     /**
      *
      * Method to check if the document required fields have been defined for the
          creation.
      *
      * @param Document $document The document model object
      * @return bool
      */
    private function validateDocument(Document $document)
    {
        if ($document->type == null) {
            return false;
        }

        if ($document->user_id == null) {
            return false;
        }

        return true;
    }

    /**
      *
      * Method to check if the conditions registration document required fields have
           been defined for the creation.
     *
     * @param Document $document The document model object
     * @return bool
     */
    private function validateDocumentConditionsRegistration(Document $document)
    {
        if ($document->package_number == null) {
            return false;
        }

        if ($document->signature_base64 == null) {
            return false;
        }

        return true;
    }
}
```

Listing 4 – ./Sources/app/Controllers/DogController.php

```
<?php
/**
 * DogController.php
 *
 * Controller of the Dog model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAODog;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\Models\Dog;
use App\System\Constants;

class DogController {

    private DAODog $DAODog;
    private DAOUser $DAOUser;


    /**
     *
     * Constructor of the DogController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAODog = new DAODog($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all dogs in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllDogs()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allDogs = $this->DAODog->findAll();

        return ResponseController::successfulRequest($allDogs);
    }

    /**
     *
     * Method to return a dog in JSON format.
     *
     * @param int $id The dog identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getDog(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $dog = $this->DAODog->find($id);

        if (is_null($dog)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($dog);
    }

    /**
     *
     * Method to create a dog.
     *
     * @param Dog $dog The dog model object
     * @return string The status and the body in JSON format of the response
     */
    public function createDog(Dog $dog)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        if (!$this->validateDog($dog)) {
            return ResponseController::unprocessableEntityResponse();
        }

        $user = $this->DAOUser->find($dog->user_id);

        if (is_null($user)) {
            return ResponseController::notFoundResponse();
        }

        $this->DAODog->insert($dog);

        return ResponseController::successfulCreatedRessource();
    }

    /**
     *
     * Method to update a dog.
     *
     * @param Dog $dog The dog model object
     * @return string The status and the body in JSON format of the response
     */
    public function updateDog(Dog $dog)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $actualDog = $this->DAODog->find($dog->id);

        if (is_null($actualDog)) {
            return ResponseController::notFoundResponse();
        }

        $actualDog->name = $dog->name ?? $actualDog->name;
        $actualDog->breed = $dog->breed ?? $actualDog->breed;
        $actualDog->sex = $dog->sex ?? $actualDog->sex;
        $actualDog->picture_serial_id = $dog->picture_serial_id ?? $actualDog->
            picture_serial_id;
        $actualDog->chip_id = $dog->chip_id ?? $actualDog->chip_id;

        $this->DAODog->update($actualDog);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to delete a dog.
     *
     * @param int  $id The dog identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteDog(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $dog = $this->DAODog->find($id);

        if (is_null($dog)) {
            return ResponseController::notFoundResponse();
        }

        if (!is_null($dog->picture_serial_id)) {
            $filename = HelperController::getDefaultDirectory()."storage/app/
                dog_picture/".$dog->picture_serial_id.".jpeg";
            if (file_exists($filename)) {
                unlink($filename);
            }
        }

        $this->DAODog->delete($dog);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to upload a dog picture.
     *
     * @return string The status and the body in JSON format of the response
     */
    public function uploadDogPicture()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        if (!isset($_FILES["dog_picture"]) || !is_uploaded_file($_FILES["
            dog_picture"]["tmp_name"]) || !isset($_POST["dog_id"])) {
            return ResponseController::unprocessableEntityResponse();
        }

        $dog = $this->DAODog->find($_POST["dog_id"]);

        if (is_null($dog)) {
            return ResponseController::notFoundResponse();
        }

        switch ($_FILES["dog_picture"]["type"]) {
            case Constants::IMAGE_TYPE_PNG:
                HelperController::pngTojpegConverter($_FILES["dog_picture"]["
                    tmp_name"]);
                break;
            case Constants::IMAGE_TYPE_JPEG:
                break;
            default:
                return ResponseController::imageFileFormatProblem();
                break;
        }

        $tmp_file = $_FILES["dog_picture"]["tmp_name"];
        $img_name = HelperController::generateRandomString();
        $upload_file = HelperController::getDefaultDirectory()."storage/app/
            dog_picture/".$img_name.".jpeg";

        if (!is_null($dog->picture_serial_id)) {
            $filename = HelperController::getDefaultDirectory()."storage/app/
                dog_picture/".$dog->picture_serial_id.".jpeg";
            if (file_exists($filename)) {
                unlink($filename);
            }
        }

        if (!move_uploaded_file($tmp_file,$upload_file)) {
            return ResponseController::uploadFailed();
        }

        $dog->picture_serial_id = $img_name;

        $this->DAODog->update($dog);
```

```php

        return ResponseController::successfulRequest();
    }

    /**
     *
     * Method to download a dog picture.
     *
     * @param string  $serial_id The serial_id of the dog picture
     * @return string The status and the body in JSON format of the response
     */
    public function downloadDogPicture(string $serial_id)
    {
        if(is_null($this->DAODog->findBySerialId($serial_id))){
            return ResponseController::notFoundResponse();
        }

        $image = HelperController::getDefaultDirectory()."storage/app/dog_picture/"
            .$serial_id.".jpeg";

        $imageInfo = getimagesize($image);

        header("Content-Type: " . $imageInfo["mime"]);
        header("Content-Length: " . filesize($image));
        readfile($image);

        return ResponseController::successfulRequest();
    }

     /**
     *
     * Method to check if the dog required fields have been defined for the
         creation.
     *
     * @param Dog $dog The dog model object
     * @return bool
     */
    private function validateDog(Dog $dog)
    {
        if ($dog->name == null) {
            return false;
        }

        if ($dog->breed == null) {
            return false;
        }

        if ($dog->sex == null) {
            return false;
        }

        if ($dog->user_id == null) {
            return false;
        }

        return true;
    }
}
```

Listing 5 – ./Sources/app/Controllers/HelperController.php

```php
<?php
/**
 * HelperController.php
 *
 * Controller allowing the use of help functions.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;
use PHPMailer\PHPMailer\PHPMailer;
use Dompdf\Dompdf;

class HelperController {

     /**
     *
     * Method to check if a date has the right format (YYYY-MM-DD).
     *
     * @param string $date Date to check
     * @return bool
     */
    public static function validateDateFormat(string $date)
    {
        if (!preg_match("/^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])$/",
            $date)) {
            return false;
        }

        $elements = explode("-",$date);

        if (!checkdate($elements[1],$elements[2],$elements[0])) {
            return false;
        }

        return true;
    }

    /**
     *
     * Method to check if a date has the right format (YYYY-MM-DD).
     *
     * @param string $date Date to check
     * @return bool
     */
    public static function validateDateTimeFormat(string $datetime, string $format
        = 'Y-m-d H:i:s')
    {
        if (!preg_match("/^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])
            (0[1-9]|1[0-9]|2[0-4]):(0[0-9]|[1-5][0-9]):(0[0-9]|[1-5][0-9])$/",
            $datetime)) {
            return false;
        }

        $date = \DateTime::createFromFormat($format, $datetime);
        if ($date && $date->format($format) != $datetime) {
            return false;
        }
        return true;
    }

    /**
     *
     * Method to check if a time has the right format (HH:MM:SS).
     *
     * @param string $time Time to check
     * @return bool
     */
    public static function validateTimeFormat(string $time)
    {
        if (!preg_match("/^(0[1-9]|1[0-9]|2[0-4]):(0[0-9]|[1-5][0-9])
           :(0[0-9]|[1-5][0-9])$/",$time)) {
            return false;
        }

        return true;
    }

    /**
     *
     * Method to check if a code day has the right format (1-6).
     *
     * @param string $time code day to check
     * @return bool
     */
    public static function validateCodeDayFormat(string $code_day)
    {
        if (!preg_match("/^[1-7]$/",$code_day)) {
            return false;
        }

        return true;
    }

    /**
     *
     * Method to check if the first time is smaller or equal to the second one.
     *
     * @param string $firsttime First time to check
     * @param string $secondtime Second time to check
     * @return bool
     */
    public static function validateChornologicalTime(string $firsttime, string
       $secondtime)
    {
        $firstdate = strtotime($firsttime);
        $seconddate = strtotime($secondtime);

        if ($seconddate < $firstdate) {
           return false;
        }

        return true;
    }

    /**
     *
     * Method to generate an api token.
     *
     * @return string The api token
     */
    public static function generateApiToken()
    {
        return md5(microtime());
    }

    /**
     *
     * Method to generate a random string.
     *
     * @return string The random string
     */
    public static function generateRandomString() {
        $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
            ';
        $password = array();
        $alphaLength = strlen($alphabet) - 1;
        for ($i = 0; $i < 8; $i++) {
            $n = rand(0, $alphaLength);
            $password[] = $alphabet[$n];
        }
        return implode($password);
    }

    /**
     *
     * Method to check if a document type has the right value (
         conditions_inscription,poster).
     *
     * @param string $document_type document type to check
     * @return bool
     */
    public static function validateDocumentTypeFormat(string $document_type)
    {
        if ($document_type == 'conditions_inscription' || $document_type == 'poster
            ') {
            return true;
        }

        return false;
    }

    /**
     *
     * Method to check if a email has the right format.
     *
     * @param string $email email  to check
     * @return bool
     */
    public static function validateEmailFormat(string $email)
    {
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            return false;
        }

        return true;
    }

    /**
     *
     * Method to check if a package number of conditions of registration document
         has the right format.
     *
     * @param int $package_number Package number to check
     * @return bool
     */
    public static function validatePackageNumber(int $package_number)
    {
        if ($package_number < 1 || $package_number > 5) {
            return false;
        }

        return true;
    }

    /**
     *
     * Method to load the email HTML CSS template.
     *
     * @param string $title The title of the mail
     * @param string $content The content of the mail
     * @param string $importantContent The important content of the mail
     * @return string
     */
    public static function loadMailTemplate(string $title, string $content, string
        $importantContent = null)
    {
        $html ='<!DOCTYPE html>
        <html lang="fr">
        <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width,initial-scale=1">
          <meta name="x-apple-disable-message-reformatting">
          <title></title>
          <style>
            table, td, div, h1, p {font-family: Arial, sans-serif;}
          </style>
        </head>
        <body style="margin:0;padding:0;">
          <table role="presentation" style="width:100%;border-collapse:collapse;
              border:0;border-spacing:0;background:#ffffff;">
            <tr>
              <td align="center" style="padding:0;">
                <table role="presentation" style="width:602px;border-collapse:
                    collapse;border:1px solid #cccccc;border-spacing:0;text-align:
                    left;">
                  <tr>
                    <td align="center" style="padding:40px 0 30px 0;">
                      <img src="cid:logo-image" alt="" width="300" style="height:
                          auto;display:block;" />
                    </td>
                  </tr>
                  <tr>
                    <td style="padding:36px 30px 42px 30px;">
                      <table role="presentation" style="width:100%;border-collapse:collapse;border:0;border-spacing:0;">
                        <tr>
                          <td style="padding:0 0 36px 0;color:#153643;">
                            <h1 style="font-size:24px;margin:0 0 20px 0;font-family:Arial,sans-serif;">'.$title.'</h1>
                            <p style="text-align: justify;margin:0 0 12px 0;font-size:16px;line-height:24px;font-family:Arial,sans-serif;">'.$content.'</p>
                            <h3 style="font-size:18px;margin:0 0 20px 0;font-family:Arial,sans-serif;">'.$importantContent.'</h3>
                          </td>
                        </tr>
                      </table>
                    </td>
                  </tr>
                  <tr>
                    <td style="padding:30px;background:#3ea3d8;">
                      <table role="presentation" style="width:100%;border-collapse:collapse;border:0;border-spacing:0;font-size:9px;font-family:Arial,sans-serif;">
                        <tr>
                          <td style="padding:0;width:50%;" align="left">
                            <p style="margin:0;font-size:14px;line-height:16px;font-family:Arial,sans-serif;color:#ffffff;">
                              &reg; Douceur de Chien 2021<br/>
                              <a href="https://boreljaquet.ch/" style="color:#ffffff;text-decoration:underline;">Site web Douceur de Chien</a>
                            </p>
                          </td>
                          <td style="padding:0;width:50%;" align="right">
                            <table role="presentation" style="border-collapse:collapse;border:0;border-spacing:0;">
                              <tr>
                                <td style="padding:0 0 0 10px;width:38px;">
                                  <a href="http://www.twitter.com/" style="color:#ffffff;"><img src="https://assets.codepen.io/210284/tw_1.png" alt="Twitter" width="38" style="height:auto;display:block;border:0;" /></a>
                                </td>
                                <td style="padding:0 0 0 10px;width:38px;">
                                  <a href="http://www.facebook.com/" style="color:#ffffff;"><img src="https://assets.codepen.io/210284/fb_1.png" alt="Facebook" width="38" style="height:auto;display:block;border:0;" /></a>
                                </td>
                              </tr>
                            </table>
                          </td>
                        </tr>
                      </table>
                    </td>
                  </tr>
                </table>
                </td>
            </tr>
        </table>
    </body>
    </html>';
    return $html;
}

/**
 *
 * Method to send an email.
 *
 * @param string $message Message of the mail
 * @param string $subject Subject of the mail
 * @param string $emailRecipient Recipient's email address
 * @param string $importantMessage Important message if there is one
 * @param string $attachmentFilePath Path of the file to attach if there is one
 * @return void
 */
public static function sendMail(string $message, string $subject,string
    $emailRecipient, string $importantMessage = null, string
    $attachmentFilePath = null)
{
$mail = new PHPMailer(true);
$body = HelperController::loadMailTemplate($subject,$message,$importantMessage)
    ;
try {
    //Server settings
    $mail->SMTPDebug = SMTP::DEBUG_SERVER;
    $mail->Host       = getenv('SMTP_HOST');
    $mail->SMTPAuth   = true;
    $mail->Username   = getenv('SMTP_USERNAME');
    $mail->Password   = getenv('SMTP_PASSWORD');
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port       = 587;

    //Recipients
    $mail->setFrom('noreply@douceurdechien.com', 'Douceur de Chien');
    $mail->addAddress($emailRecipient);

    //Attachments
    if (!is_null($attachmentFilePath)) {
        $mail->addAttachment($attachmentFilePath);
    }

    //Content
    $mail->isHTML(true);
    $mail->CharSet = 'UTF-8';
    $mail->Encoding = 'base64';
    $mail->Subject = $subject;
    $mail->AddEmbeddedImage(HelperController::getDefaultDirectory()."resources/
        image/logo_douceur_de_chien.png", "logo-image", "logo_douceur_de_chien.
        png");
    $mail->Body = $body;

    $mail->send();
} catch (Exception $e) {
    echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
}
```

```php
    }

    /**
     *
     * Method to convert a png image to jpeg image.
     *
     * @return void
     */
    public static function pngTojpegConverter(string $filename)
    {
        $image = imagecreatefrompng ($filename);
        imagejpeg($image, $filename);
        imagedestroy($image);
    }

    /**
     *
     * Method to store a conditions of registration.
     *
     * @param string $filename The filename of conditions of registration
     * @param string $package_number The package number of conditions of
        registration
     * @param string $date The date of creation of conditions of registration
     * @param string $signature_base64 The signature image in base64 of conditions
        of registration
     * @param string $userfirstname The user firstname of owner of conditions of
        registration
     * @param string $userlastname The user lastname of owner of conditions of
        registration
     *
     * @return void
     */
    public static function storeConditionsRegistration(string $filename,int
       $package_number,string $date, string $signature_base64,string
       $userfirstname, string $userlastname)
    {
        $dompdf = new DOMPDF();

        ob_start();
        include HelperController::getDefaultDirectory()."resources/template/
            conditions_registration.php";
        $contents = ob_get_clean();

        $dompdf->loadHtml($contents);
        $dompdf->render();
        $output = $dompdf->output();

        file_put_contents(HelperController::getDefaultDirectory()."storage/app/
            conditions_registration/".$filename.".pdf", $output);
    }

    /**
     *
     * Method to check if the client-side Google reCAPTCHA is valid or not
     *
     * @param string $userResponseToken The user response token provided by the
        client-side integration.
     * @link https://developers.google.com/recaptcha/docs/verify#api-request
     *
     * @return bool
     */
    public static function reCAPTCHAvalidate(string $userResponseToken)
    {
      $curl = curl_init();

      curl_setopt($curl,CURLOPT_URL, "https://www.google.com/recaptcha/api/
          siteverify");
      curl_setopt($curl, CURLOPT_POST,1);
      curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query(array('secret' =>
          getenv('GOOGLE_RECAPTCHA_SECRET_KEY'),'response' => $userResponseToken)))
          ;
      curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

      $response = json_decode(curl_exec($curl));

      return $response->success;
    }

    /**
     *
     * Method to send an email with the appointment information and an ics file (
        iCalendar) as an attachment.
     *
     * @param Datetime $start_datetime Start date of the appointment in an Datetime
          instance
     * @param Datetime $end_datetime End date of the appointment in an Datetime
        instance
     * @param string $educator_fullname The first and last name of the educator
     * @param string $customer_email The customer's email address
     * @param string $filename The name of the ics file to send
     * @link https://datatracker.ietf.org/doc/html/rfc5545
     *
     * @return void
     */
    public static function sendMailWithICSFile(\Datetime $start_datetime, \Datetime
         $end_datetime, string $educator_fullname, string $customer_email, string
        $filename)
    {
      ob_start();
      include HelperController::getDefaultDirectory()."resources/template/
          iCalendar_appointment.php";
      $contents = ob_get_clean();

      $temp = tmpfile();
      fwrite($temp, $contents);

      $tmpfile_path = stream_get_meta_data($temp)['uri'];
      $new_tmpfile_path = sys_get_temp_dir(). '\\' . $filename;
      rename($tmpfile_path, $new_tmpfile_path);

      $message = "Bonjour et merci de faire confiance à la société Douceur de Chien
          ,
      vous venez de prendre un rendez-vous le ".$start_datetime->format("d-m-Y")."
      de ".$start_datetime->format("H\h")." à ".$end_datetime->format("H\h")." avec
      l'éducateur canin ".$educator_fullname.".
      Vous trouverez ci-joint un fichier ICS permettant de planifier le rendez-vous
      dans votre calendrier Microsoft, Google ou Apple. Vous pouvez également accé
          der aux informations
      de ce rendez-vous depuis votre compte.";

      HelperController::sendMail($message,"Un nouveau rendez-vous a été planifié",
          $customer_email,null,$new_tmpfile_path);

      unlink($new_tmpfile_path);
    }

    /**
     *
     * Method to return the default directory of the API.
     *
     * @return string The default directory
     */
    public static function getDefaultDirectory(){
        return $_SERVER["DOCUMENT_ROOT"]."/bj-travail-diplome-2021/api_rest/";
    }
}
```

Listing 6 – ./Sources/app/Controllers/ResponseController.php

```
<?php
/**
 * ResponseController.php
 *
 * Controller allowing to return different type of HTTP response.
 *
 * @link https://developer.mozilla.org/fr/docs/Web/HTTP/Status
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;


class ResponseController {



    /**
     *
     * Method to return the error message in case of undefined authorization header
         .
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function notFoundAuthorizationHeader()
    {
        $response['status_code_header'] = 'HTTP/1.1 401 Unauthorized';
        $response['body'] = json_encode([
            'error' => "L'en-tête d'autorisation n'est pas défini."
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of no access permission.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function unauthorizedUser()
    {
        $response['status_code_header'] = 'HTTP/1.1 403 Forbidden';
        $response['body'] = json_encode([
            'error' => "Vous n'avez pas les permissions."
        ]);
        return $response;
    }
    /**
     *
     * Method to return the error message in case of not found response.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function notFoundResponse()
    {
        $response['status_code_header'] = 'HTTP/1.1 404 Not Found';
        $response['body'] = json_encode([
            'error' => "Le serveur n'a pas trouvé la ressource demandée."
        ]);

        return $response;
    }

    /**
     *
     * Method to return the GET success message.
     *
     * @param array $result The associative array containing all the result rows
     * @return string The status and the body in JSON format of the response
     */
    public static function successfulRequest($result = null)
    {
        $response = array();

        $response['status_code_header'] = 'HTTP/1.1 200 OK';
        $response['body'] = (!is_null($result)) ? json_encode($result) : null;
        return $response;
    }

    /**
     *
     * Method to return the GET success message for data that does not need to be
         json encoded.
     *
     * @param array $result The associative array containing all the result rows
     * @return string The status and the body in JSON format of the response
     */
    public static function successfulRequestWithoutJson($result)
    {
        $response['status_code_header'] = 'HTTP/1.1 200 OK';
        $response['body'] = $result;
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid attributes.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function unprocessableEntityResponse()
    {
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Attributs invalides.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of a permanent calendar already
        created.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function permanentScheduleAlreadyExist()
    {
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Un calendrier permanent a déjà été créé.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the POST success message.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function successfulCreatedRessource()
    {
        $response['status_code_header'] = 'HTTP/1.1 201 Created';
        $response['body'] = null;
        return $response;
    }

    /**
     *
     * Method to return the POST success message.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function successfulCreatedRessourceWithJson($result)
    {
        $response['status_code_header'] = 'HTTP/1.1 201 Created';
        $response['body'] = json_encode($result);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid date format.
     *
     * @return string The status and the body in JSON format of the response
```

```php
     */
    public static function invalidDateFormat()
    {
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Format de date invalide => (YYYY-MM-DD).'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid time format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidTimeFormat()
    {
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Format de temps invalide => (HH:MM:SS).'
        ]);
        return $response;
    }

     /**
     *
     * Method to return the error message in case of invalid datetime format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidDateTimeFormat()
    {
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Format de date invalide => (YYYY-MM-DD HH:MM:SS).'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of date overlap problem.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function dateOverlapProblem(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Les dates chevauchent d\'autres dates déjà existantes.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of time overlap problem.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function timeOverlapProblem(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Les horaires chevauchent d\'autres horaires déjà existants.
                '
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of chronological problem between
         two times.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function chronologicalDateProblem(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'La date ou l\'heure de début est plus récente que la date
                ou l\'heure de fin.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid code day format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidCodeDayFormat(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Format de jour invalide => (1 jusqu\'à 7, dimanche = 1).'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid document type format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidDocumentTypeFormat(){
        $response['status_code_header'] = 'HTTP/1.1 415 Unsupported Media Type';
        $response['body'] = json_encode([
            'error' => 'Type de document invalide => (conditions_inscription,poster
                ).'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid email format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidEmailFormat(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Format d\'adresse email invalide.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid login.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidLogin(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Identifiants de connexion invalides.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of upload failed.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function uploadFailed(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Échec d\'upload.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid image file format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function imageFileFormatProblem(){
        $response['status_code_header'] = 'HTTP/1.1 415 Unsupported Media Type';
        $response['body'] = json_encode([
            'error' => 'Format d\'image par pris en charge.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid image file format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function documentTypeNotPdfProblem(){
        $response['status_code_header'] = 'HTTP/1.1 415 Unsupported Media Type';
        $response['body'] = json_encode([
            'error' => 'Format de document pas pris en charge => Format disponible
                PDF.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid package number of
         conditions of registration document.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function packageNumberFormatProblem(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Numéro de Forfait invalide => (1 jusqu\'à 5).'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid appointment
         informations.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidAppointment(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Impossible de prendre ce rendez-vous.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message if the email address already exists.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function emailAlreadyExist(){
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Un compte avec cette adresse e-mail existe déjà.'
        ]);
        return $response;
    }

    /**
     *
     * Method to return the error message in case of invalid captcha.
     *
     * @return string The status and the body in JSON format of the response
     */
    public static function invalidCaptcha()
```

```php
    {
        $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
        $response['body'] = json_encode([
            'error' => 'Captcha invalide.'
        ]);
        return $response;
    }


}
```

Listing 7 – ./Sources/app/Controllers/ScheduleOverrideController.php

```php
<?php
/**
 * ScheduleOverrideController.php
 *
 * Controller of the ScheduleOverride model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAOScheduleOverride;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\DataAccessObject\DAOTimeSlot;
use App\Models\ScheduleOverride;
use App\System\Constants;

class ScheduleOverrideController {

    private DAOScheduleOverride $DAOScheduleOverride;
    private DAOTimeSlot $DAOTimeSlot;
    private DAOUser $DAOUser;

    /**
     *
     * Constructor of the ScheduleOverrideController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAOScheduleOverride = new DAOScheduleOverride($db);
        $this->DAOTimeSlot = new DAOTimeSlot($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all schedule overrides and their time slots in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllScheduleOverrides()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allScheduleOverrides = $this->DAOScheduleOverride->findAll(false,$userAuth
            ->id);

        foreach ($allScheduleOverrides as $scheduleOverride) {
            $scheduleOverride->timeSlots = $this->DAOTimeSlot->
                findAllByIdScheduleOverride(false, $userAuth->id, $scheduleOverride
                ->id);
        }

        return ResponseController::successfulRequest($allScheduleOverrides);
    }

    /**
     *
     * Method to return a schedule override in JSON format.
     *
     * @param int  $id The schedule override identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getScheduleOverride(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $scheduleOverride = $this->DAOScheduleOverride->find($id,$userAuth->id);

        if (is_null($scheduleOverride)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($scheduleOverride);
    }

    /**
     *
     * Method to create a schedule override.
     *
     * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
     * @return string The status and the body in JSON format of the response
     */
    public function createScheduleOverride(ScheduleOverride $scheduleOverride)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $scheduleOverride->id_educator = $userAuth->id;

        if (!$this->validateScheduleOverride($scheduleOverride)) {
            return ResponseController::unprocessableEntityResponse();
        }

        if (!HelperController::validateDateFormat($scheduleOverride->
            date_schedule_override)) {
            return ResponseController::invalidDateFormat();
        }

        if ($this->DAOScheduleOverride->findExistence($scheduleOverride)) {
            return ResponseController::dateOverlapProblem();
        }

        $this->DAOScheduleOverride->insert($scheduleOverride);

        return ResponseController::successfulCreatedRessource();
    }

    /**
     *
     * Method to delete a schedule override.
     *
     * @param int  $id The schedule override identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteScheduleOverride(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $scheduleOverride = $this->DAOScheduleOverride->find($id,$userAuth->id);

        if (is_null($scheduleOverride)) {
            return ResponseController::notFoundResponse();
        }

        $this->DAOScheduleOverride->delete($scheduleOverride);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to check if the schedule override required fields have been defined
        for the creation.
     *
     * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
     * @return bool
     */
    private function validateScheduleOverride(ScheduleOverride $scheduleOverride)
    {
        if ($scheduleOverride->date_schedule_override == null) {
            return false;
        }

        return true;
    }
}
```

Listing 8 – ./Sources/app/Controllers/TimeSlotController.php

```
<?php
/**
 * TimeSlotController.php
 *
 * Controller of the TimeSlot model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAOTimeSlot;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\DataAccessObject\DAOScheduleOverride;
use App\DataAccessObject\DAOWeeklySchedule;
use App\Models\TimeSlot;
use App\System\Constants;

class TimeSlotController {

    private DAOTimeSlot $DAOTimeSlot;
    private DAOWeeklySchedule $DAOWeeklySchedule;
    private DAOScheduleOverride $DAOScheduleOverride;
    private DAOUser $DAOUser;


    /**
     *
     * Constructor of the TimeSlotController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAOTimeSlot = new DAOTimeSlot($db);
        $this->DAOWeeklySchedule = new DAOWeeklySchedule($db);
        $this->DAOScheduleOverride = new DAOScheduleOverride($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all time slots in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllTimeSlots()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
           ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allTimeSlots = $this->DAOTimeSlot->findAll(false,$userAuth->id);

        return ResponseController::successfulRequest($allTimeSlots);
    }

    /**
     *
     * Method to return a time slot in JSON format.
     *
     * @param int  $id The time slot identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getTimeSlot(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $timeSlot = $this->DAOTimeSlot->find($id,$userAuth->id);

        if (is_null($timeSlot)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($timeSlot);
    }

    /**
     *
     * Method to create a time slot.
     *
     * @param TimeSlot $timeSlot The timeSlot model object
     * @return string The status and the body in JSON format of the response
     */
    public function createTimeSlot(TimeSlot $timeSlot)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $timeSlot->id_educator = $userAuth->id;

        if (!$this->validateTimeSlot($timeSlot)) {
            return ResponseController::unprocessableEntityResponse();
        }

        if (!HelperController::validateCodeDayFormat($timeSlot->code_day)) {
            return ResponseController::invalidCodeDayFormat();
        }

        if (!HelperController::validateTimeFormat($timeSlot->time_start) || !
            HelperController::validateTimeFormat($timeSlot->time_end) ) {
            return ResponseController::invalidTimeFormat();
        }

        if (!HelperController::validateChornologicalTime($timeSlot->time_start,
            $timeSlot->time_end)) {
            return ResponseController::chronologicalDateProblem();
        }

        if ($this->DAOTimeSlot->findOverlapInWeeklySchedule($timeSlot) || $this->
            DAOTimeSlot->findOverlapInScheduleOverride($timeSlot))
        {
            return ResponseController::timeOverlapProblem();
        }

        $this->DAOTimeSlot->insert($timeSlot);

        return ResponseController::successfulCreatedRessource();
    }

    /**
     *
     * Method to delete a time slot.
     *
     * @param int  $id The time slot identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteTimeSlot(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $timeSlot = $this->DAOTimeSlot->find($id,$userAuth->id);

        if (is_null($timeSlot)) {
            return ResponseController::notFoundResponse();
        }

        $this->DAOTimeSlot->delete($timeSlot);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to return all available valid time slots with their corresponding
         dates in JSON format.
     *
     * @param int  $idUser The edcuator user identifier
     * @return string The status and the body in json format of the response
     */
    public function getPlanningTimeSlots(int $idUser)
    {
        $educator = $this->DAOUser->find($idUser);

        if (is_null($educator) || $educator->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::notFoundResponse();
        }

        $planning = $this->DAOTimeSlot->findPlanningForEducator($idUser);

        return ResponseController::successfulRequest($planning);
    }

    /**
     *
     * Method to check if the time slot required fields have been defined for the
        creation.
     *
     * @param TimeSlot $timeSlot The timeslot model object
     * @return bool
     */
    private function validateTimeSlot(TimeSlot $timeSlot)
    {
        if ($timeSlot->code_day == null) {
            return false;
        }

        if ($timeSlot->time_start == null) {
            return false;
        }

        if ($timeSlot->time_end == null) {
            return false;
        }

        if (($timeSlot->id_weekly_schedule != null) xor ($timeSlot->
            id_schedule_override == null)) {
            return false;
        }

        $result = null;

        if ($timeSlot->id_weekly_schedule != null) {
            $result = $this->DAOWeeklySchedule->find($timeSlot->id_weekly_schedule,
                 $timeSlot->id_educator);
        }

        if ($timeSlot->id_schedule_override != null) {
            $result = $this->DAOScheduleOverride->find($timeSlot->
                id_schedule_override, $timeSlot->id_educator);
        }

        if ($result == null) {
            return false;
        }

        return true;
    }
}
```

Listing 9 – ./Sources/app/Controllers/UserController.php

```
<?php
/**
 * UserController.php
 *
 * Controller of the User model.
 *
```

```php
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAOUser;
use App\DataAccessObject\DAODog;
use App\DataAccessObject\DAODocument;
use App\DataAccessObject\DAOAppointment;
use app\Models\User;
use App\Controllers\ResponseController;
use App\System\Constants;

class UserController {

    private DAOUser $DAOUser;
    private DAODog $DAODog;
    private DAODocument $DAODocument;
    private DAOAppointment $DAOAppointment;

    /**
     *
     * Constructor of the UserController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAOUser = new DAOUser($db);
        $this->DAODog = new DAODog($db);
        $this->DAODocument = new DAODocument($db);
        $this->DAOAppointment = new DAOAppointment($db);
    }

    /**
     *
     * Method to return all users in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllCustomerUsers()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $api_token = $headers['Authorization'];

        $userAuth = $this->DAOUser->findByApiToken($api_token);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allCustomerUsers = $this->DAOUser->findAll(Constants::USER_CODE_ROLE);

        foreach ($allCustomerUsers as $customerUser) {
            $customerUser->dogs = $this->DAODog->findByUserId($customerUser->id);
        }

        return ResponseController::successfulRequest($allCustomerUsers);
    }

    /**
     *
     * Method to return all educator users in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllEducatorUsers()
    {
        $allEducatorUsers = $this->DAOUser->findAllEducators();

        return ResponseController::successfulRequest($allEducatorUsers);
    }

    /**
     *
     * Method to return a user in JSON format.
     *
     * @param int $id The user identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getUser(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $user = $this->DAOUser->find($id);

        if (is_null($user)) {
            return ResponseController::notFoundResponse();
        }

        $dogs = $this->DAODog->findByUserId($user->id);
        $documents = $this->DAODocument->findByUserId($user->id);
        $appointments = $this->DAOAppointment->findByUserIdForAdmin($user->id);
        $user->dogs = $dogs;
        $user->documents = $documents;
        $user->appointments = $appointments;

        return ResponseController::successfulRequest($user);
    }

    /**
```

```
     *
     * Method to create a user.
     *
     * @param User $user The user model object
     * @param string $reCAPTCHAuserResponseToken The user response token provided
        by the reCAPTCHA client-side integration.
     * @return string The status and the body in JSON format of the response
     */
    public function createUser(User $user, string $reCAPTCHAuserResponseToken =
        null)
    {
        $user->api_token = HelperController::generateApiToken();
        $user->code_role = Constants::USER_CODE_ROLE;

        if (!$this->validateUser($user)) {
            return ResponseController::unprocessableEntityResponse();
        }

        if (!HelperController::validateEmailFormat($user->email)) {
            return ResponseController::invalidEmailFormat();
        }

        if (!is_null($this->DAOUser->findUserByEmail($user->email))) {
            return ResponseController::emailAlreadyExist();
        }

        if ($user->password_hash != null) {
            $user->password_hash = password_hash($user->password_hash,
                PASSWORD_DEFAULT);
            if (is_null($reCAPTCHAuserResponseToken)) {
                return ResponseController::unprocessableEntityResponse();
            }
            if (!HelperController::reCAPTCHAvalidate($reCAPTCHAuserResponseToken))
                {
                return ResponseController::invalidCaptcha();
            }
        }
        else{
            $random_password = HelperController::generateRandomString();
            $user->password_hash = password_hash($random_password,PASSWORD_DEFAULT)
                ;
        }

        $this->DAOUser->insert($user);

        if (isset($random_password)) {
            HelperController::sendMail("Bonjour et merci de faire confiance à la
                société Douceur de Chien, vous trouverez plus bas votre mot de
                passe généré aléatoirement afin d'accéder à votre compte. Toutefois
                , il est fortement conseillé de modifier votre mot de passe une
                fois connecté.","Création de votre compte Douceur de Chien",$user->
                email,$random_password);
        }

        $newUser = $this->DAOUser->findUserByEmail($user->email);

        $result = array();
        $result["api_token"] = $user->api_token;
        $result["user_id"] = $newUser->id;
```

```php
        $result["code_role"] = $user->code_role;
        return ResponseController::successfulCreatedRessourceWithJson($result);
    }

    /**
     *
     * Method to update a user.
     *
     * @param User $user The user model object
     * @return string The status and the body in JSON format of the response
     */
    public function updateUser(User $user)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $actualUser = $this->DAOUser->find($user->id);

        if (is_null($actualUser)) {
            return ResponseController::notFoundResponse();
        }

        $actualUser->email = $user->email ?? $actualUser->email;
        $actualUser->firstname = $user->firstname ?? $actualUser->firstname;
        $actualUser->lastname = $user->lastname ?? $actualUser->lastname;
        $actualUser->phonenumber = $user->phonenumber ?? $actualUser->phonenumber;
        $actualUser->address = $user->address ?? $actualUser->address;

        if (!HelperController::validateEmailFormat($actualUser->email)) {
            return ResponseController::invalidEmailFormat();
        }

        $this->DAOUser->update($actualUser);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to delete a user.
     *
     * @param int  $id The user identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteUser(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $user = $this->DAOUser->find($id);

        if (is_null($user)) {
            return ResponseController::notFoundResponse();
        }

        $dogs = $this->DAODog->findByUserId($user->id);

        foreach ($dogs as $dog) {

            if (!is_null($dog->picture_serial_id)) {

                $filename = HelperController::getDefaultDirectory()."storage/app/
                    dog_picture/".$dog->picture_serial_id.".jpeg";

                if (file_exists($filename)) {
                    unlink($filename);
                }
            }
        }

        $documents = $this->DAODocument->findByUserId($user->id);

        foreach ($documents as $document) {

            if (!is_null($document->document_serial_id)) {

                if ($document->type == Constants::
                    DOCUMENT_TYPE_CONDTIONS_OF_REGISTRATION) {

                    $filename = HelperController::getDefaultDirectory()."storage/
                        app/conditions_registration/".$document->document_serial_id
                        .".pdf";
                }
                else{
                    $filename = HelperController::getDefaultDirectory()."storage/
                        app/pdf/".$document->document_serial_id.".pdf";
                }

                if (file_exists($filename)) {

                    unlink($filename);
                }
            }
        }

        $appointments = $this->DAOAppointment->findByUserId($user->id);

        foreach ($appointments as $appointment) {

            if (!is_null($appointment->note_graphical_serial_id)) {

                $filename = HelperController::getDefaultDirectory()."storage/app/
                    graphical_note/".$appointment->note_graphical_serial_id.".png";

                if (file_exists($filename)) {
                    unlink($filename);
                }
            }
        }

        $this->DAOUser->delete($user);

        return ResponseController::successfulRequest(null);
    }

    /**
     *
     * Method to get the api token of a user by logging in.
     *
     * @param User $user The user model object
     * @return string The status and the body in JSON format of the response
     */
    public function connection(User $user)
    {
        if (is_null($user->email) || is_null($user->password_hash)) {
            return ResponseController::unprocessableEntityResponse();
        }

        $userAuth = $this->DAOUser->findUserByEmail($user->email);

        if (is_null($userAuth)) {
            return ResponseController::invalidLogin();
        }

        if (!password_verify($user->password_hash,$userAuth->password_hash)) {
            return ResponseController::invalidLogin();
        }

        $userAuth->api_token = HelperController::generateApiToken();

        $this->DAOUser->update($userAuth);

        $result = array();
        $result["api_token"] = $userAuth->api_token;
        $result["user_id"] = $userAuth->id;
        $result["code_role"] = $userAuth->code_role;
        return ResponseController::successfulRequest($result);
    }

    /**
     *
     * Method to return all information of the currently logged in user in JSON
        format.
     *
     * @return string The status and the body in JSON format of the response
     */
    public function getMyInformations()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth)) {
            return ResponseController::notFoundResponse();
        }
        $dogs = $this->DAODog->findByUserId($userAuth->id);
        $documents = $this->DAODocument->findByUserId($userAuth->id);
        $appointments = $this->DAOAppointment->findByUserId($userAuth->id);
        $userAuth->dogs = $dogs;
        $userAuth->documents = $documents;
        $userAuth->appointments = $appointments;


        return ResponseController::successfulRequest($userAuth);
    }

    /**
     *
     * Method to update the password of the currently logged user.
     *
     * @param User $user The user model object
     * @return string The status and the body in JSON format of the response
     */
    public function updateMyPassword(User $user)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth)) {
            return ResponseController::notFoundResponse();
        }

        $userAuth->password_hash = password_hash($user->password_hash,
            PASSWORD_DEFAULT) ?? $userAuth->password_hash;

        $this->DAOUser->update($userAuth);

        return ResponseController::successfulRequest(null);
    }

     /**
     *
     * Method to check if the user required fields have been defined for the
         creation.
     *
     * @param User $user The user model object
     * @return bool
     */
    private function validateUser(User $user)
    {
        if ($user->email == null) {
            return false;
        }

        if ($user->firstname == null) {
            return false;
        }

        if ($user->lastname == null) {
            return false;
        }

        if ($user->phonenumber == null) {
            return false;
        }

        if ($user->address == null) {
            return false;
        }

        return true;
    }
}
```

Listing 10 – ./Sources/app/Controllers/WeeklyScheduleController.php

```
<?php
/**
 * WeeklyScheduleController.php
 *
 * Controller of the WeeklySchedule model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

namespace App\Controllers;

use App\DataAccessObject\DAOWeeklySchedule;
use App\DataAccessObject\DAOUser;
use App\Controllers\ResponseController;
use App\Controllers\HelperController;
use App\DataAccessObject\DAOTimeSlot;
use App\Models\WeeklySchedule;
use App\System\Constants;

class WeeklyScheduleController {

    private DAOWeeklySchedule $DAOWeeklySchedule;
    private DAOTimeSlot $DAOTimeSlot;
    private DAOUser $DAOUser;

    /**
     *
     * Constructor of the WeeklyScheduleController object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->DAOWeeklySchedule = new DAOWeeklySchedule($db);
        $this->DAOTimeSlot = new DAOTimeSlot($db);
        $this->DAOUser = new DAOUser($db);
    }

    /**
     *
     * Method to return all weekly schedules and their time slots in JSON format.
     *
     * @return string The status and the body in json format of the response
     */
    public function getAllWeeklySchedules()
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $allWeeklySchedule = $this->DAOWeeklySchedule->findAll(false,$userAuth->id)
            ;

        foreach ($allWeeklySchedule as $weeklySchedule) {
            $weeklySchedule->timeSlots = $this->DAOTimeSlot->
                findAllByIdWeeklySchedule(false, $userAuth->id, $weeklySchedule->id
                );
        }

        return ResponseController::successfulRequest($allWeeklySchedule);
    }

    /**
     *
     * Method to return a weekly schedule in JSON format.
     *
     * @param int  $id The weekly schedule identifier
     * @return string The status and the body in JSON format of the response
     */
    public function getWeeklySchedule(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $weeklySchedule = $this->DAOWeeklySchedule->find($id,$userAuth->id);

        if (is_null($weeklySchedule)) {
            return ResponseController::notFoundResponse();
        }

        return ResponseController::successfulRequest($weeklySchedule);
    }

    /**
     *
     * Method to create a weekly schedule.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return string The status and the body in JSON format of the response
     */
    public function createWeeklySchedule(WeeklySchedule $weeklySchedule)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $weeklySchedule->id_educator = $userAuth->id;

        if (!$this->validateWeeklySchedule($weeklySchedule)) {
            return ResponseController::unprocessableEntityResponse();
        }

        if (!HelperController::validateDateFormat($weeklySchedule->date_valid_from)
            ) {
            return ResponseController::invalidDateFormat();
        }

        if (!is_null($weeklySchedule->date_valid_to)) {
            if (!HelperController::validateDateFormat($weeklySchedule->
                date_valid_to)) {
                return ResponseController::invalidDateFormat();
            }
            if (!HelperController::validateChornologicalTime($weeklySchedule->
                date_valid_from,$weeklySchedule->date_valid_to)) {
                return ResponseController::chronologicalDateProblem();
            }
        }
        else{
            if ($this->DAOWeeklySchedule->findActifPermanentSchedule(
                $weeklySchedule)) {
```

```
                return ResponseController::permanentScheduleAlreadyExist();
            }
        }

        if ($this->DAOWeeklySchedule->findOverlap($weeklySchedule)) {
            return ResponseController::dateOverlapProblem();
        }

        $this->DAOWeeklySchedule->insert($weeklySchedule);

        return ResponseController::successfulCreatedRessource();
    }

    /**
     *
     * Method to delete a weekly schedule.
     *
     * @param int  $id The weekly schedule identifier
     * @return string The status and the body in JSON format of the response
     */
    public function deleteWeeklySchedule(int $id)
    {
        $headers = apache_request_headers();

        if (!isset($headers['Authorization'])) {
            return ResponseController::notFoundAuthorizationHeader();
        }

        $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

        if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
        }

        $weeklySchedule = $this->DAOWeeklySchedule->find($id,$userAuth->id);

        if (is_null($weeklySchedule)) {
            return ResponseController::notFoundResponse();
        }

        $this->DAOWeeklySchedule->delete($weeklySchedule);

        return ResponseController::successfulRequest(null);
    }

   /**
     *
     * Method to check if the weekly schedule required fields have been defined for
        the creation.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return bool
     */
    private function validateWeeklySchedule(WeeklySchedule $weeklySchedule)
    {
        if ($weeklySchedule->date_valid_from == null) {
            return false;
        }

        return true;
    }
}
```

Listing 11 – ./Sources/app/DataAccessObject/DAOAbsence.php

```php
<?php
/**
 * DAOAbsence.php
 *
 * Data access object of the absence table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

use App\Models\Absence;

class DAOAbsence {

    private $db = null;

    /**
     *
     * Constructor of the DAOAbsence object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all the absences of the database in an array of absence
         objects.
     *
     * @param bool $isDeleted  Bool to define whether to search for existing or
         deleted absences
     * @param int $idEducator The educator identifier
     * @return Absence[] A Absence object array
     */
    public function findAll(bool $isDeleted,int $idEducator)
    {
        $statement ="
        SELECT id, date_absence_from, date_absence_to, description, id_educator
        FROM absence
        WHERE is_deleted= :DELETED
        AND id_educator = :ID_EDUCATOR
        ORDER BY date_absence_from";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->bindParam(':DELETED', $isDeleted, \PDO::PARAM_BOOL);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $absenceArray = array();

            foreach ($results as $result) {
                $absence = new Absence();
                $absence->id = $result["id"];
                $absence->date_absence_from = $result["date_absence_from"];
                $absence->date_absence_to = $result["date_absence_to"];
                $absence->description = $result["description"];
                $absence->id_educator = $result["id_educator"];
                array_push($absenceArray,$absence);
            }

            return $absenceArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return an absence from the database in a absence model object.
     *
     * @param int $id The absence identifier
     * @param int $idEducator The educator identifier
     * @return Absence A Absence model object containing all the result rows of the
        query
     */
    public function find(int $id,int $idEducator)
    {
        $statement = "
        SELECT id, date_absence_from, date_absence_to, description, id_educator
        FROM absence
        WHERE id = :ID_ABSENCE
        AND is_deleted = 0
        AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_ABSENCE', $id, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->execute();

            $absence = new Absence();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $absence->id = $result["id"];
                $absence->date_absence_from = $result["date_absence_from"];
                $absence->date_absence_to = $result["date_absence_to"];
                $absence->description = $result["description"];
                $absence->id_educator = $result["id_educator"];
            }
            else{
                $absence = null;
            }

            return $absence;
        } catch (\PDOException $e) {
```

```php
                exit($e->getMessage());
            }
        }

         /**
         *
         * Method to insert a absence in the database.
         *
         * @param Absence $absence The absence model object
         * @return int The number of rows affected by the insert
         */
        public function insert(Absence $absence)
        {
            $statement = "
            INSERT INTO absence (date_absence_from, date_absence_to, description,
                id_educator ,is_deleted)
            VALUES(STR_TO_DATE(:DATE_ABSENCE_FROM, \"%Y-%m-%d\"), STR_TO_DATE(:
                DATE_ABSENCE_TO, \"%Y-%m-%d\"), :DESCRIPTION,:ID_EDUCATOR ,0)";

            try {
                $statement = $this->db->prepare($statement);
                $statement->bindParam(':DATE_ABSENCE_FROM', $absence->date_absence_from
                    , \PDO::PARAM_STR);
                $statement->bindParam(':DATE_ABSENCE_TO', $absence->date_absence_to, \
                    PDO::PARAM_STR);
                $statement->bindParam(':DESCRIPTION', $absence->description, \PDO::
                    PARAM_STR);
                $statement->bindParam(':ID_EDUCATOR', $absence->id_educator, \PDO::
                    PARAM_INT);
                $statement->execute();
                return $statement->rowCount();
            } catch (\PDOException $e) {
                exit($e->getMessage());
            }
        }

        /**
         *
         * Method to update a absence in the database.
         *
         * @param Absence $absence The absence model object
         * @return int The number of rows affected by the update
         */
        public function update(Absence $absence)
        {
            $statement = "
            UPDATE absence
            SET date_absence_from = STR_TO_DATE(:DATE_ABSENCE_FROM, \"%Y-%m-%d\"),
            date_absence_to = STR_TO_DATE(:DATE_ABSENCE_TO, \"%Y-%m-%d\"),
            description = :DESCRIPTION
            WHERE id = :ID_ABSENCE";

            try {
                $statement = $this->db->prepare($statement);
                $statement->bindParam(':DATE_ABSENCE_FROM', $absence->date_absence_from
                    , \PDO::PARAM_STR);
                $statement->bindParam(':DATE_ABSENCE_TO', $absence->date_absence_to, \
                    PDO::PARAM_STR);
                $statement->bindParam(':DESCRIPTION', $absence->description, \PDO::
```

```
                PARAM_STR);
            $statement->bindParam(':ID_ABSENCE', $absence->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a absence in the database.
     *
     * @param Absence $absence The absence model object
     * @return int The number of rows affected by the delete
     */
    public function delete(Absence $absence)
    {
        $statement = "
        UPDATE absence
        SET is_deleted = 1
        WHERE id = :ID_ABSENCE";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_ABSENCE', $absence->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 12 – ./Sources/app/DataAccessObject/DAOAppointment.php

```
<?php
/**
 * DAOAppointment.php
 *
 * Data access object of the appointment table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

use App\Models\Appointment;

class DAOAppointment {

    private $db = null;

    /**
     *
     * Constructor of the DAOAppointment object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all the appointments of the database in an array of
         appointment objects.
     *
     * @param int $userCustomerId The customer identifier
     * @param int $userEducatorId The educator identifier
     * @return Appointment[] A Appointment object array
     */
    public function findAll(int $userCustomerId = null, int $userEducatorId = null)
    {
        $statement = "
        SELECT id, datetime_appointment,duration_in_hour, note_text,
        note_graphical_serial_id, summary, datetime_deletion,
        user_id_customer,user_id_educator,user_id_deletion
        FROM appointment
                WHERE (user_id_customer = :ID_USER_CUSTOMER OR user_id_educator = :
                    ID_USER_EDUCATOR)
                AND datetime_deletion IS NULL
                AND user_id_deletion IS NULL";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER_CUSTOMER', $userCustomerId, \PDO::
                PARAM_INT);
            $statement->bindParam(':ID_USER_EDUCATOR', $userEducatorId, \PDO::
                PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $appointmentArray = array();

            foreach ($results as $result) {
                $appointment = new Appointment();
                $appointment->id = $result["id"];
                $appointment->datetime_appointment = $result["datetime_appointment"
                    ];
                $appointment->duration_in_hour = $result["duration_in_hour"];
                $appointment->note_text = $result["note_text"];
                $appointment->note_graphical_serial_id = $result["
                    note_graphical_serial_id"];
                $appointment->summary = $result["summary"];
                $appointment->user_id_customer = $result["user_id_customer"];
                $appointment->user_id_educator = $result["user_id_educator"];
                array_push($appointmentArray,$appointment);
            }

            return $appointmentArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a appointment slot from the database in a appointment model
        object.
     *
     * @param int $id The appointment identifier
     * @param int $idEducator The educator identifier
     * @return Appointment A Appointment model object containing all the result
        rows of the query
     */
    public function find(int $id)
    {
        $statement = "
        SELECT id, datetime_appointment,duration_in_hour, note_text,
        note_graphical_serial_id, summary, datetime_deletion,
        user_id_customer,user_id_educator,user_id_deletion
        FROM appointment
        WHERE id = :ID_APPOINTMENT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_APPOINTMENT', $id, \PDO::PARAM_INT);
            $statement->execute();

            $appointment = new Appointment();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $appointment->id = $result["id"];
                $appointment->datetime_appointment = $result["datetime_appointment"
                    ];
                $appointment->duration_in_hour = $result["duration_in_hour"];
                $appointment->note_text = $result["note_text"];
                $appointment->note_graphical_serial_id = $result["
                    note_graphical_serial_id"];
                $appointment->summary = $result["summary"];
                $appointment->user_id_customer = $result["user_id_customer"];
                $appointment->user_id_educator = $result["user_id_educator"];
            }
            else{
                $appointment = null;
            }

            return $appointment;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return all appointment from the database with the user id in an
        array of appointment objects without the administrator data.
     *
     * @param int $userId The user identifier
     * @return Appointment[] A Appointment object array
     */
    public function findByUserId(int $userId)
    {
        $statement = "
        SELECT id, datetime_appointment,duration_in_hour, summary,
        user_id_customer,user_id_educator
        FROM appointment
        WHERE user_id_customer = :ID_USER
        AND datetime_deletion IS NULL
        AND user_id_deletion IS NULL
        ORDER BY datetime_appointment DESC";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $appointmentArray = array();

            foreach ($results as $result) {
                $appointment = new Appointment();
                $appointment->id = $result["id"];
                $appointment->datetime_appointment = $result["datetime_appointment"
                    ];
                $appointment->duration_in_hour = $result["duration_in_hour"];
                $appointment->summary = $result["summary"];
                $appointment->user_id_customer = $result["user_id_customer"];
                $appointment->user_id_educator = $result["user_id_educator"];
                array_push($appointmentArray,$appointment);
            }

            return $appointmentArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return all appointment from the database with the user id in an
         array of appointment objects.
     *
     * @param int $userId The user identifier
     * @return Appointment[] A Appointment object array
     */
    public function findByUserIdForAdmin(int $userId)
    {
        $statement = "
        SELECT id, datetime_appointment,duration_in_hour,note_text,
            note_graphical_serial_id, summary,
        user_id_customer,user_id_educator
        FROM appointment
        WHERE user_id_customer = :ID_USER
        AND datetime_deletion IS NULL
        AND user_id_deletion IS NULL
        ORDER BY datetime_appointment DESC";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $appointmentArray = array();

            foreach ($results as $result) {
                $appointment = new Appointment();
                $appointment->id = $result["id"];
                $appointment->datetime_appointment = $result["datetime_appointment"
                    ];
                $appointment->duration_in_hour = $result["duration_in_hour"];
                $appointment->note_text = $result["note_text"];
                $appointment->note_graphical_serial_id = $result["
                    note_graphical_serial_id"];
                $appointment->summary = $result["summary"];
                $appointment->user_id_customer = $result["user_id_customer"];
                $appointment->user_id_educator = $result["user_id_educator"];
                array_push($appointmentArray,$appointment);
            }

            return $appointmentArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a appointment from the database from his graphical note
         serial id.
     *
     * @param string $serial_id The graphical note serial id
     * @return Appointment A Appointment model object containing all the result
         rows of the query
     */
    public function findBySerialId(string $serial_id)
    {
        $statement = "
        SELECT id, datetime_appointment,duration_in_hour, note_text,
        note_graphical_serial_id, summary, datetime_deletion,
        user_id_customer,user_id_educator,user_id_deletion
        FROM appointment
        WHERE note_graphical_serial_id = :SERIAL_ID";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
            $statement->execute();

            $appointment = new Appointment();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $appointment->id = $result["id"];
                $appointment->datetime_appointment = $result["datetime_appointment"
                    ];
                $appointment->duration_in_hour = $result["duration_in_hour"];
                $appointment->note_text = $result["note_text"];
                $appointment->note_graphical_serial_id = $result["
                    note_graphical_serial_id"];
                $appointment->summary = $result["summary"];
                $appointment->user_id_customer = $result["user_id_customer"];
                $appointment->user_id_educator = $result["user_id_educator"];
            }
            else{
                $appointment = null;
            }

            return $appointment;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to insert a appointment in the database.
     *
     * @param Appointment $appointment The absence model object
     * @return int The number of rows affected by the insert
     */
    public function insert(Appointment $appointment)
    {
        $statement = "
        INSERT INTO appointment (datetime_appointment,duration_in_hour,
        user_id_customer,user_id_educator)

        VALUES(:DATETIME_APPOINTMENT, :DURATION_IN_HOUR,
        :USER_ID_CUSTOMER,:USER_ID_EDUCATOR)";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATETIME_APPOINTMENT', $appointment->
                datetime_appointment, \PDO::PARAM_STR);
            $statement->bindParam(':DURATION_IN_HOUR', $appointment->
                duration_in_hour, \PDO::PARAM_INT);
            $statement->bindParam(':USER_ID_CUSTOMER', $appointment->
                user_id_customer, \PDO::PARAM_INT);
            $statement->bindParam(':USER_ID_EDUCATOR', $appointment->
                user_id_educator, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a appointment in the database.
     *
     * @param Appointment $appointment The absence model object
     * @return int The number of rows affected by the update
     */
    public function update(Appointment $appointment)
    {
        $statement = "
        UPDATE appointment
        SET note_text = :NOTE_TEXT,
        note_graphical_serial_id = :NOTE_GRAPHICAL_SERIAL_ID,
        summary = :SUMMARY
        WHERE id = :ID_APPOINTMENT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':NOTE_TEXT', $appointment->note_text, \PDO::
                PARAM_STR);
            $statement->bindParam(':NOTE_GRAPHICAL_SERIAL_ID', $appointment->
                note_graphical_serial_id, \PDO::PARAM_STR);
            $statement->bindParam(':SUMMARY', $appointment->summary, \PDO::
                PARAM_STR);
            $statement->bindParam(':ID_APPOINTMENT', $appointment->id, \PDO::
                PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a appointment in the database.
     *
     * @param int $userId The user who deletes the appointment
     * @param int $id The appointment identifier
     * @return int The number of rows affected by the delete
     */
    public function delete(int $id,int $idUser)
    {
        $statement = "
        UPDATE appointment
        SET datetime_deletion = NOW(),
        user_id_deletion = :ID_USER
        WHERE id = :ID_APPOINTMENT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $idUser, \PDO::PARAM_INT);
            $statement->bindParam(':ID_APPOINTMENT', $id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 13 – ./Sources/app/DataAccessObject/DAODocument.php

```
<?php
/**
 * DAODocument.php
 *
 * Data access object of the document table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;
```

```php

use App\Models\Document;

class DAODocument {

    private \PDO $db;

    /**
     *
     * Constructor of the DAODocument object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all the documents of the database in an array of document
         objects.
     *
     * @return Document[] A Document object array
     */
    public function findAll()
    {
        $statement = "
        SELECT id, document_serial_id, type, user_id
        FROM document";

        try {
            $statement = $this->db->prepare($statement);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $documentArray = array();

            foreach ($results as $result) {
                $document = new Document();
                $document->id = $result["id"];
                $document->document_serial_id = $result["document_serial_id"];
                $document->type = $result["type"];
                $document->user_id = $result["user_id"];
                array_push($documentArray,$document);
            }

            return $documentArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }


    /**
     *
     * Method to return a document from the database in a document model object.
     *
     * @param int $id The document identifier
     * @return Document A Document model object containing all the result rows of
        the query
     */
    public function find(int $id)
    {
        $statement = "
        SELECT id, document_serial_id, type, user_id
        FROM document
        WHERE id = :ID_DOCUMENT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_DOCUMENT', $id, \PDO::PARAM_INT);
            $statement->execute();

            $document = new Document();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $document->id = $result["id"];
                $document->document_serial_id = $result["document_serial_id"];
                $document->type = $result["type"];
                $document->user_id = $result["user_id"];
            }
            else{
                $document = null;
            }

            return $document;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return all documents from the database with the user id in an
        array of documents objects.
     *
     * @param int $userId The user identifier
     * @return Document[] A Document object array
     */
    public function findByUserId(int $userId)
    {
        $statement = "
        SELECT id, document_serial_id, type, user_id
        FROM document
        WHERE user_id = :ID_USER";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $documentArray = array();

            foreach ($results as $result) {
                $document = new Document();
                $document->id = $result["id"];
                $document->document_serial_id = $result["document_serial_id"];
                $document->type = $result["type"];
                $document->user_id = $result["user_id"];
                array_push($documentArray,$document);
            }

            return $documentArray;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a document from the database with the serial id in a
         document model object.
     *
     * @param string $serial_id The serial id of the document
     * @return Document A Document model object containing all the result rows of
         the query
     */
    public function findBySerialId(string $serial_id)
    {
        $statement = "
        SELECT id, document_serial_id, type, user_id
        FROM document
        WHERE document_serial_id= :SERIAL_ID";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
            $statement->execute();

            $document = new Document();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $document->id = $result["id"];
                $document->document_serial_id = $result["document_serial_id"];
                $document->type = $result["type"];
                $document->user_id = $result["user_id"];
            }
            else{
                $document = null;
            }

            return $document;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a document from the database with the user id and serial id
          in a document model object.
     *
     * @param int $userId The user identifier
     * @param string $serial_id The serial id of the document
     * @return Document A Document model object containing all the result rows of
        the query
     */
    public function findByUserIdAndSerialId(int $userId,string $serial_id)
    {
        $statement = "
        SELECT id, document_serial_id, type, user_id
        FROM document
        WHERE user_id = :ID_USER
        AND document_serial_id= :SERIAL_ID";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
            $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
            $statement->execute();

            $document = new Document();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $document->id = $result["id"];
                $document->document_serial_id = $result["document_serial_id"];
                $document->type = $result["type"];
                $document->user_id = $result["user_id"];
            }
            else{
                $document = null;
            }

            return $document;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to insert a document in the database.
     *
     * @param Document $document The document model object
     * @return int The number of rows affected by the insert
     */
    public function insert(Document $document)
    {
        $statement = "
        INSERT INTO document (document_serial_id, type,user_id)
        VALUES(:DOCUMENT_SERIAL_ID, :TYPE, :USER_ID)";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DOCUMENT_SERIAL_ID', $document->
                document_serial_id, \PDO::PARAM_STR);
            $statement->bindParam(':TYPE', $document->type, \PDO::PARAM_STR);
            $statement->bindParam(':USER_ID', $document->user_id, \PDO::PARAM_STR);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a document in the database.
     *
     * @param Document $document The document model object
     * @return int The number of rows affected by the update
     */
    public function update(Document $document)
    {
        $statement = "
        UPDATE document
        SET document_serial_id = :DOCUMENT_SERIAL_ID,
        type = :TYPE,
        user_id = :USER_ID
        WHERE id = :ID_DOCUMENT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DOCUMENT_SERIAL_ID', $document->
                document_serial_id, \PDO::PARAM_STR);
            $statement->bindParam(':TYPE', $document->type, \PDO::PARAM_STR);
            $statement->bindParam(':USER_ID', $document->user_id, \PDO::PARAM_STR);
            $statement->bindParam(':ID_DOCUMENT', $document->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a document in the database.
     *
     * @param Document $document The document model object
     * @return int The number of rows affected by the delete
     */
    public function delete(Document $document)
    {
        $statement = "
        DELETE FROM document
        WHERE id = :ID_DOCUMENT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_DOCUMENT', $document->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
```

```
}
```

Listing 14 – ./Sources/app/DataAccessObject/DAODog.php

```php
<?php
/**
 * DAODog.php
 *
 * Data access object of the dog table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

use App\Models\Dog;


class DAODog {

    private \PDO $db;

    /**
     *
     * Constructor of the DAODog object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all dogs from the database in an array of dog objects.
     *
     * @return Dog[] A Dog object array
     */
    public function findAll()
    {
        $statement = "
        SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
        FROM dog";

        try {
            $statement = $this->db->prepare($statement);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $dogArray = array();

            foreach ($results as $result) {
                $dog = new Dog();
                $dog->id = $result["id"];
                $dog->name = $result["name"];
                $dog->breed = $result["breed"];
                $dog->sex = $result["sex"];
                $dog->picture_serial_id = $result["picture_serial_id"];
                $dog->chip_id = $result["chip_id"];
                $dog->user_id = $result["user_id"];
                array_push($dogArray,$dog);
            }

            return $dogArray;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }


    /**
     *
     * Method to return a dog from the database in a dog model object.
     *
     * @param int $id The dog identifier
     * @return Dog A Dog model object containing all the result rows of the query
     */
    public function find(int $id)
    {
        $statement = "
        SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
        FROM dog
        WHERE id = :ID_DOG";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_DOG', $id, \PDO::PARAM_INT);
            $statement->execute();

            $dog = new Dog();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $dog->id = $result["id"];
                $dog->name = $result["name"];
                $dog->breed = $result["breed"];
                $dog->sex = $result["sex"];
                $dog->picture_serial_id = $result["picture_serial_id"];
                $dog->chip_id = $result["chip_id"];
                $dog->user_id = $result["user_id"];
            }
            else{
                $dog = null;
            }

            return $dog;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return all dogs from the database with the user id in an array of
        dog objects.
     *
     * @param int $userId The user identifier
     * @return Dog[] A Dog object array
     */
    public function findByUserId(int $userId)
    {
        $statement = "
        SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
        FROM dog
        WHERE user_id = :ID_USER";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $dogArray = array();

            foreach ($results as $result) {
                $dog = new Dog();
                $dog->id = $result["id"];
                $dog->name = $result["name"];
                $dog->breed = $result["breed"];
                $dog->sex = $result["sex"];
                $dog->picture_serial_id = $result["picture_serial_id"];
                $dog->chip_id = $result["chip_id"];
                $dog->user_id = $result["user_id"];
                array_push($dogArray,$dog);
            }

            return $dogArray;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a dog from the database from his picture serial id.
     *
     * @param string $serial_id The dog picture serial id
     * @return Dog A Dog model object containing all the result rows of the query
     */
    public function findBySerialId(string $serial_id)
    {
        $statement = "
        SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
        FROM dog
        WHERE picture_serial_id = :SERIAL_ID";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
            $statement->execute();

            $dog = new Dog();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $dog->id = $result["id"];
                $dog->name = $result["name"];
                $dog->breed = $result["breed"];
                $dog->sex = $result["sex"];
                $dog->picture_serial_id = $result["picture_serial_id"];
                $dog->chip_id = $result["chip_id"];
                $dog->user_id = $result["user_id"];
            }
            else{
                $dog = null;
            }

            return $dog;

        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to insert a dog in the database.
     *
     * @param Dog $dog The dog model object
     * @return int The number of rows affected by the insert
     */
    public function insert(Dog $dog)
    {
        $statement = "
        INSERT INTO dog (name, breed, sex, picture_serial_id, chip_id, user_id)
        VALUES(:NAME, :BREED, :SEX, :PICTURE_SERIAL_ID, :CHIP_ID, :USER_ID)";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':NAME', $dog->name, \PDO::PARAM_STR);
            $statement->bindParam(':BREED', $dog->breed, \PDO::PARAM_STR);
            $statement->bindParam(':SEX', $dog->sex, \PDO::PARAM_STR);
            $statement->bindParam(':PICTURE_SERIAL_ID', $dog->picture_serial_id, \
                PDO::PARAM_STR);
            $statement->bindParam(':CHIP_ID', $dog->chip_id, \PDO::PARAM_STR);
            $statement->bindParam(':USER_ID', $dog->user_id, \PDO::PARAM_STR);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a dog in the database.
     *
     * @param Dog $dog The dog model object
     * @return int The number of rows affected by the update
     */
    public function update(Dog $dog)
    {
        $statement = "
        UPDATE dog
        SET name = :NAME,
        breed = :BREED,
        sex = :SEX,
        picture_serial_id = :PICTURE_SERIAL_ID,
        chip_id = :CHIP_ID
        WHERE id = :ID_DOG";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':NAME', $dog->name, \PDO::PARAM_STR);
            $statement->bindParam(':BREED', $dog->breed, \PDO::PARAM_STR);
            $statement->bindParam(':SEX', $dog->sex, \PDO::PARAM_STR);
            $statement->bindParam(':PICTURE_SERIAL_ID', $dog->picture_serial_id, \
                PDO::PARAM_STR);
            $statement->bindParam(':CHIP_ID', $dog->chip_id, \PDO::PARAM_STR);
            $statement->bindParam(':ID_DOG', $dog->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a dog in the database.
     *
     * @param Dog $dog The dog model object
     * @return int The number of rows affected by the delete
     */
    public function delete(Dog $dog)
    {
        $statement = "
        DELETE FROM dog
        WHERE id = :ID_DOG";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_DOG', $dog->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 15 – ./Sources/app/DataAccessObject/DAOScheduleOverride.php

```
<?php
/**
 * DAOScheduleOverride.php
 *
 * Data access object of the schedule_override table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

```

```
use App\Models\ScheduleOverride;

class DAOScheduleOverride {

    private $db = null;

    /**
     *
     * Constructor of the DAOScheduleOverride object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all the schedule override of the database in an array of
         scheduleoverride objects.
     *
     * @param bool $isDeleted  Bool to define whether to search for existing or
         deleted scheduleoverride
     * @param int $idEducator The educator identifier
     * @return ScheduleOverride[] A ScheduleOverride object array
     */
    public function findAll(bool $deleted, int $idEducator)
    {
        $statement ="
        SELECT id, date_schedule_override, id_educator
        FROM schedule_override
        WHERE is_deleted= :DELETED
        AND id_educator = :ID_EDUCATOR
        ORDER BY date_schedule_override";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $scheduleOverrideArray = array();

            foreach ($results as $result) {
                $scheduleOverride = new ScheduleOverride();
                $scheduleOverride->id = $result["id"];
                $scheduleOverride->date_schedule_override = $result["
                    date_schedule_override"];
                $scheduleOverride->id_educator = $result["id_educator"];
                array_push($scheduleOverrideArray,$scheduleOverride);
            }

            return $scheduleOverrideArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a schedule override from the database in a scheduleoverride
        model object.
     *
     * @param int $id The scheduleoverride identifier
     * @param int $idEducator The educator identifier
     * @return ScheduleOverride A ScheduleOverride model object containing all the
        result rows of the query
     */
    public function find(int $id, int $idEducator)
    {
        $statement = "
        SELECT id, date_schedule_override, id_educator
        FROM schedule_override
        WHERE id = :ID_SCHEDULE_OVERRIDE
        AND is_deleted = 0
        AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $id, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->execute();

            $scheduleOverride = new ScheduleOverride();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $scheduleOverride->id = $result["id"];
                $scheduleOverride->date_schedule_override = $result["
                    date_schedule_override"];
                $scheduleOverride->id_educator = $result["id_educator"];
            }
            else{
                $scheduleOverride = null;
            }

            return $scheduleOverride;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

     /**
     *
     * Method to insert a schedule override in the database.
     *
     * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
     * @return int The number of rows affected by the insert
     */
    public function insert(ScheduleOverride $scheduleOverride)
    {
        $statement = "
        INSERT INTO schedule_override (date_schedule_override, id_educator,
            is_deleted)
        VALUES(STR_TO_DATE(:DATE_SCHEDULE_OVERRIDE, \"%Y-%m-%d\"), :ID_EDUCATOR, 0)
            ";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE_SCHEDULE_OVERRIDE', $scheduleOverride->
                date_schedule_override, \PDO::PARAM_STR);
            $statement->bindParam(':ID_EDUCATOR', $scheduleOverride->id_educator, \
                PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a schedule override in the database.
     *
     * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
     * @return int The number of rows affected by the update
     */
    public function update(ScheduleOverride $scheduleOverride)
    {
        $statement = "
        UPDATE schedule_override
        SET date_schedule_override = STR_TO_DATE(:DATE_SCHEDULE_OVERRIDE, \"%Y-%m-%
            d\")
        WHERE id = :ID_SCHEDULE_OVERRIDE";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE_SCHEDULE_OVERRIDE', $scheduleOverride->
                date_schedule_override, \PDO::PARAM_STR);
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $scheduleOverride->id, \
                PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

     /**
     *
     * Method to delete a schedule override in the database.
     *
     * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
     * @return int The number of rows affected by the delete
     */
    public function delete(ScheduleOverride $scheduleOverride)
    {
        $statement = "
        UPDATE schedule_override
        SET is_deleted = 1
        WHERE id = :ID_SCHEDULE_OVERRIDE";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $scheduleOverride->id, \
                PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

     /**
     *
     * Method to check if a date has not already been defined for the same educator
         in the database.
     *
     * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
     * @return array The associative array containing all the result rows of the
         query
     */
    public function findExistence(ScheduleOverride $scheduleOverride)
    {
        $statement = "
        SELECT *
        FROM schedule_override
        WHERE date_schedule_override = STR_TO_DATE(:DATE_SCHEDULE_OVERRIDE, \"%Y-%m
            -%d\")
        AND is_deleted = 0
        AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE_SCHEDULE_OVERRIDE', $scheduleOverride->
                date_schedule_override, \PDO::PARAM_STR);
            $statement->bindParam(':ID_EDUCATOR', $scheduleOverride->id_educator, \
                PDO::PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 16 – ./Sources/app/DataAccessObject/DAOTimeSlot.php

```
<?php
/**
 * DAOTimeSlot.php
 *
 * Data access object of the time_slot table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

use App\Models\TimeSlot;

class DAOTimeSlot {

    private $db = null;

    /**
     *
     * Constructor of the DAOTimeSlot object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all the time slot for an educator of the database in an
         array of timeslot objects.
     *
     * @param bool $isDeleted  Bool to define whether to search for existing or
         deleted timeslot
     * @param int $idEducator The educator identifier
     * @return TimeSlot[] A ScheduleOverride object array
     */
    public function findAll(bool $deleted,int $idEducator)
    {
        $statement ="
        SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
        FROM time_slot
        WHERE is_deleted= :DELETED
        AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $timeSlotArray = array();

            foreach ($results as $result) {
                $timeSlot = new TimeSlot();
                $timeSlot->id = $result["id"];
                $timeSlot->code_day = $result["code_day"];
                $timeSlot->time_start = $result["time_start"];
                $timeSlot->time_end = $result["time_end"];
                $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
                $timeSlot->id_schedule_override = $result["id_schedule_override"];
                $timeSlot->id_educator = $result["id_educator"];
                array_push($timeSlotArray,$timeSlot);
            }

            return $timeSlotArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
```

```
     *
     * Method to return all the time slot of the database in an array of timeslot
         objects from his weekly schedule identifier.
     *
     * @param bool $isDeleted  Bool to define whether to search for existing or
         deleted timeslot
     * @param int $idEducator The educator identifier
     * @param int $idWeeklySchedule The weekly schedule identifier
     * @return TimeSlot[] A ScheduleOverride object array
     */
    public function findAllByIdWeeklySchedule(bool $deleted,int $idEducator, int
        $idWeeklySchedule)
    {
        $statement ="
        SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
        FROM time_slot
        WHERE is_deleted= :DELETED
        AND id_educator = :ID_EDUCATOR
        AND id_weekly_schedule = :ID_WEEKLY_SCHEDULE
        ORDER BY code_day, time_start";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $idWeeklySchedule, \PDO::
                PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $timeSlotArray = array();

            foreach ($results as $result) {
                $timeSlot = new TimeSlot();
                $timeSlot->id = $result["id"];
                $timeSlot->code_day = $result["code_day"];
                $timeSlot->time_start = $result["time_start"];
                $timeSlot->time_end = $result["time_end"];
                $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
                $timeSlot->id_schedule_override = $result["id_schedule_override"];
                $timeSlot->id_educator = $result["id_educator"];
                array_push($timeSlotArray,$timeSlot);
            }

            return $timeSlotArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return all the time slot of the database in an array of timeslot
         objects from his schedule override identifier.
     *
     * @param bool $isDeleted  Bool to define whether to search for existing or
         deleted timeslot
     * @param int $idEducator The educator identifier
```

```php
     * @param int $idScheduleOverride The schedule override identifier
     * @return TimeSlot[] A ScheduleOverride object array
     */
    public function findAllByIdScheduleOverride(bool $deleted,int $idEducator, int
        $idScheduleOverride)
    {
        $statement ="
        SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
        FROM time_slot
        WHERE is_deleted= :DELETED
        AND id_educator = :ID_EDUCATOR
        AND id_schedule_override = :ID_SCHEDULE_OVERRIDE
        ORDER BY time_start";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $idScheduleOverride, \
                PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $timeSlotArray = array();

            foreach ($results as $result) {
                $timeSlot = new TimeSlot();
                $timeSlot->id = $result["id"];
                $timeSlot->code_day = $result["code_day"];
                $timeSlot->time_start = $result["time_start"];
                $timeSlot->time_end = $result["time_end"];
                $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
                $timeSlot->id_schedule_override = $result["id_schedule_override"];
                $timeSlot->id_educator = $result["id_educator"];
                array_push($timeSlotArray,$timeSlot);
            }

            return $timeSlotArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a time slot from the database in a timeslot model object.
     *
     * @param int $id The timeslot identifier
     * @param int $idEducator The educator identifier
     * @return TimeSlot A TimeSlot model object containing all the result rows of
        the query
     */
    public function find(int $id, int $idEducator)
    {
        $statement = "
        SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
        FROM time_slot
        WHERE id = :ID_TIMESLOT
        AND is_deleted = 0
        AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_TIMESLOT', $id, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->execute();

            $timeSlot = new TimeSlot();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $timeSlot->id = $result["id"];
                $timeSlot->code_day = $result["code_day"];
                $timeSlot->time_start = $result["time_start"];
                $timeSlot->time_end = $result["time_end"];
                $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
                $timeSlot->id_schedule_override = $result["id_schedule_override"];
                $timeSlot->id_educator = $result["id_educator"];
            }
            else{
                $timeSlot = null;
            }

            return $timeSlot;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

     /**
     *
     * Method to insert a time slot in the database.
     *
     * @param TimeSlot $timeSlot The timeslot model object
     * @return int The number of rows affected by the insert
     */
    public function insert(TimeSlot $timeSlot)
    {
        $statement = "
        INSERT INTO time_slot (code_day, time_start, time_end, is_deleted,
            id_weekly_schedule, id_schedule_override, id_educator)
        VALUES(:CODE_DAY, :TIME_START, :TIME_END, 0, :ID_WEEKLY_SCHEDULE, :
            ID_SCHEDULE_OVERRIDE,:ID_EDUCATOR)";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT
                );
            $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::
                PARAM_STR);
            $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR
                );
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $timeSlot->
                id_weekly_schedule, \PDO::PARAM_INT);
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $timeSlot->
                id_schedule_override, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $timeSlot->id_educator, \PDO::
                PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a time slot in the database.
     *
     * @param TimeSlot $timeSlot The timeslot model object
     * @return int The number of rows affected by the update
     */
    public function update(TimeSlot $timeSlot)
    {
        $statement = "
        UPDATE time_slot
        SET code_day = :CODE_DAY, time_start = :TIME_START, time_end = :TIME_END,
            id_weekly_schedule = :ID_WEEKLY_SCHEDULE, id_schedule_override = :
            ID_SCHEDULE_OVERRIDE
        WHERE id = :ID_TIMESLOT";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT
                );
            $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::
                PARAM_STR);
            $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR
                );
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $timeSlot->
                id_weekly_schedule, \PDO::PARAM_INT);
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $timeSlot->
                id_schedule_override, \PDO::PARAM_INT);
            $statement->bindParam(':ID_TIMESLOT', $timeSlot->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a time slot in the database.
     *
     * @param TimeSlot $timeSlot The timeslot model object
     * @return int The number of rows affected by the delete
     */
    public function delete(TimeSlot $timeSlot)
    {
        $statement = "
        UPDATE time_slot
        SET is_deleted = 1
        WHERE id = :ID_TIMESLOT";

```

```php
        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_TIMESLOT', $timeSlot->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to check if a time slot causes an overlapping problem in a weekly schedule.
     *
     * @param TimeSlot $timeSlot The timeslot model object
     * @return array The associative array containing all the result rows of the query
     */
    public function findOverlapInWeeklySchedule(TimeSlot $timeSlot)
    {
        $statement = "
        SELECT ts.code_day, ts.time_start, ts.time_end
        FROM time_slot AS ts
        LEFT JOIN weekly_schedule AS ws
        ON ts.id_weekly_schedule = ws.id
        WHERE ts.id_weekly_schedule = :ID_WEEKLY_SCHEDULE
        AND ts.is_deleted = 0
        AND ws.is_deleted = 0
        AND :TIME_START < time_end
        AND :TIME_END > time_start
        AND code_day = :CODE_DAY
        AND ts.id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT);
            $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::PARAM_STR);
            $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR);
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $timeSlot->id_weekly_schedule, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $timeSlot->id_educator, \PDO::PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to check if a time slot causes an overlapping problem in a schedule override.
     *
     * @param TimeSlot $timeSlot The timeslot model object
     * @return array The associative array containing all the result rows of the
         query
     */
    public function findOverlapInScheduleOverride(TimeSlot $timeSlot)
    {
        $statement = "
        SELECT ts.code_day, ts.time_start, ts.time_end
        FROM time_slot AS ts
        LEFT JOIN schedule_override AS so
        ON ts.id_schedule_override = so.id
        WHERE ts.id_schedule_override = :ID_SCHEDULE_OVERRIDE
        AND ts.is_deleted = 0
        AND so.is_deleted = 0
        AND :TIME_START < time_end
        AND :TIME_END > time_start
        AND code_day = :CODE_DAY
        AND ts.id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT
                );
            $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::
                PARAM_STR);
            $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR
                );
            $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $timeSlot->
                id_schedule_override, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $timeSlot->id_educator, \PDO::
                PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method for generating virtual views of the application.
     * 1. digits : Table containing the values from 0 to 9
     * 2. numbers : Table containing numbers from 0 to 999
     * 3. dates : Table containing dates from (today - 999 days) to (today + 1 year
         )
     *
     * @return void
     */
    private function generateViews()
    {
        $statement = "
        CREATE OR REPLACE VIEW digits AS
        SELECT 0 AS digit UNION ALL
        SELECT 1 UNION ALL
        SELECT 2 UNION ALL
        SELECT 3 UNION ALL
        SELECT 4 UNION ALL
        SELECT 5 UNION ALL
        SELECT 6 UNION ALL
        SELECT 7 UNION ALL
        SELECT 8 UNION ALL
        SELECT 9;

        CREATE OR REPLACE VIEW numbers AS
        SELECT ones.digit + tens.digit * 10 + hundreds.digit * 100 AS number
        FROM digits as ones, digits as tens, digits as hundreds;

        CREATE OR REPLACE VIEW dates AS
        SELECT SUBDATE(ADDDATE(CURRENT_DATE(),365), number) AS date
        FROM numbers";

        $this->db->exec($statement);
    }

    /**
     *
     * Method to retrieve the available valid time slots with their corresponding
         dates.
     *
     * @param int $idEducator The educator identifier
     * @return array The associative array containing all the result rows of the
         query
     */
    public function findPlanningForEducator(int $idEducator)
    {
        $this->generateViews();

        $statement = "
        SELECT IF(dates.date IS NOT NULL, dates.date, so.date_schedule_override) AS
             date,time_start,time_end

        FROM time_slot AS ts
        LEFT JOIN weekly_schedule AS ws
        ON ws.id = ts.id_weekly_schedule

        LEFT JOIN schedule_override AS so
        ON so.id = ts.id_schedule_override

        LEFT JOIN dates
        ON DAYOFWEEK(dates.date) = ts.code_day
        AND dates.date BETWEEN ws.date_valid_from
        AND IF(ws.date_valid_to IS NULL, DATE_ADD(NOW(), INTERVAL 365 DAY), ws.
            date_valid_to)

        WHERE ts.is_deleted = 0 AND (so.is_deleted = 0 OR ws.is_deleted = 0)
        AND ts.id_educator = :ID_EDUCATOR

        AND (SELECT COUNT(*)
        FROM absence AS ab
        WHERE ab.id_educator = :ID_EDUCATOR
        AND ab.is_deleted = 0
        AND IF(so.date_schedule_override IS NULL,dates.date,so.
            date_schedule_override) BETWEEN ab.date_absence_from AND ab.
            date_absence_to LIMIT 1) = 0

        AND (SELECT COUNT(*)
        FROM appointment AS ap
        WHERE ap.user_id_educator = :ID_EDUCATOR
        AND datetime_deletion IS NULL
        AND user_id_deletion IS NULL
        AND DATE(ap.datetime_appointment) = IF(so.date_schedule_override IS NULL,
            dates.date,so.date_schedule_override)
        AND TIME(ap.datetime_appointment) = ts.time_start
        AND TIME(ADDTIME(ap.datetime_appointment,SEC_TO_TIME(3600* ap.
            duration_in_hour))) = ts.time_end LIMIT 1) = 0

        AND IF(so.date_schedule_override IS NULL,dates.date > NOW() ,so.
            date_schedule_override >NOW())

        AND (SELECT COUNT(*)
                FROM schedule_override
                WHERE schedule_override.date_schedule_override = dates.date
                AND schedule_override.is_deleted = 0
                AND schedule_override.id_educator = :ID_EDUCATOR ) = 0

        ORDER BY date,time_start";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to check if appointment data is valid in an educator's planning.
     *
     * @param string $date The date of the appointment
     * @param string $time_start The start time of the appointment
     * @param string $time_end The end time of the appointment
     * @param int $idEducator The educator identifier
     * @return array The associative array containing all the result rows of the
         query
     */
    public function findAppointmentSlotsForEducator(string $date,string $time_start
        , string $time_end,int $idEducator)
    {
        $this->generateViews();

        $statement = "
        SELECT IF(dates.date IS NOT NULL, dates.date, so.date_schedule_override) AS
             date,time_start,time_end

        FROM time_slot AS ts
        LEFT JOIN weekly_schedule AS ws
        ON ws.id = ts.id_weekly_schedule

        LEFT JOIN schedule_override AS so
        ON so.id = ts.id_schedule_override

        LEFT JOIN dates
        ON DAYOFWEEK(dates.date) = ts.code_day
        AND dates.date BETWEEN ws.date_valid_from
        AND IF(ws.date_valid_to IS NULL, DATE_ADD(NOW(), INTERVAL 365 DAY), ws.
            date_valid_to)

        WHERE ts.is_deleted = 0 AND (so.is_deleted = 0 OR ws.is_deleted = 0)
        AND ts.id_educator = :ID_EDUCATOR


        AND (SELECT COUNT(*)
        FROM absence AS ab
        WHERE ab.id_educator = :ID_EDUCATOR
        AND IF(so.date_schedule_override IS NULL,dates.date,so.
            date_schedule_override) BETWEEN ab.date_absence_from AND ab.
            date_absence_to LIMIT 1) = 0

        AND (SELECT COUNT(*)
        FROM appointment AS ap
        WHERE ap.user_id_educator = :ID_EDUCATOR
        AND ap.user_id_deletion IS NULL
        AND ap.datetime_deletion IS NULL
        AND DATE(ap.datetime_appointment) = IF(so.date_schedule_override IS NULL,
            dates.date,so.date_schedule_override)
        AND TIME(ap.datetime_appointment) = ts.time_start
        AND TIME(ADDTIME(ap.datetime_appointment,SEC_TO_TIME(3600* ap.
            duration_in_hour))) = ts.time_end LIMIT 1) = 0

        AND IF(so.date_schedule_override IS NULL,dates.date > NOW() ,so.
            date_schedule_override >NOW())

        AND (SELECT COUNT(*)
                FROM schedule_override
                WHERE schedule_override.date_schedule_override = dates.date
                AND schedule_override.is_deleted = 0
                AND schedule_override.id_educator = :ID_EDUCATOR ) = 0

        HAVING DATE = :DATE
        AND time_start = :TIME_START
        AND time_end = :TIME_END

        ORDER BY date,time_start";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE', $date, \PDO::PARAM_STR);
            $statement->bindParam(':TIME_START', $time_start, \PDO::PARAM_STR);
            $statement->bindParam(':TIME_END', $time_end, \PDO::PARAM_STR);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 17 – ./Sources/app/DataAccessObject/DAOUser.php

```php
<?php
/**
 * DAOUser.php
 *
 * Data access object of the user table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

use App\Models\User;

class DAOUser {

    private \PDO $db;

    /**
     *
     * Constructor of the DAOUser object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all users with the selected role from the database in an
        array of user objects.
     *
     * @return User[] A User object array
     */
    public function findAll(int $code_role)
    {
        $statement = "
        SELECT id, email, firstname, lastname, phonenumber, address
        FROM user
        WHERE code_role = :CODE_ROLE";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':CODE_ROLE', $code_role, \PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $userArray = array();

            foreach ($results as $result) {
                $user = new User();
                $user->id = $result["id"];
                $user->email = $result["email"];
                $user->firstname = $result["firstname"];
                $user->lastname = $result["lastname"];
                $user->phonenumber = $result["phonenumber"];
                $user->address = $result["address"];
                array_push($userArray,$user);
            }
            return $userArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return all users with the administrator role from the database in
        an array of user objects.
     *
     * @return User[] A User object array
     */
    public function findAllEducators()
    {
        $statement = "
        SELECT id, firstname, lastname
        FROM user
        WHERE code_role = 2";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':CODE_ROLE', $code_role, \PDO::PARAM_INT);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $userArray = array();

            foreach ($results as $result) {
                $user = new User();
                $user->id = $result["id"];
                $user->firstname = $result["firstname"];
                $user->lastname = $result["lastname"];
                array_push($userArray,$user);
            }
            return $userArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a user from the database in a user model object.
     *
     * @param int $id The user identifier
     * @return User A User model object containing all the result rows of the query
     */
    public function find(int $id)
    {
        $statement = "
        SELECT id,email, firstname, lastname, phonenumber, address,api_token,
            code_role, password_hash
        FROM user
        WHERE id = :ID_USER";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $id, \PDO::PARAM_INT);
            $statement->execute();

            $user = new User();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $user->id = $result["id"];
                $user->email = $result["email"];
                $user->firstname = $result["firstname"];
                $user->lastname = $result["lastname"];
                $user->phonenumber = $result["phonenumber"];
                $user->address = $result["address"];
                $user->api_token = $result["api_token"];
                $user->code_role = $result["code_role"];
                $user->password_hash = $result["password_hash"];
            }
            else{
                $user = null;
            }

            return $user;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to insert a user in the database.
     *
     * @param User $user The user model object
     * @return int The number of rows affected by the insert
     */
    public function insert(User $user)
    {
        $statement = "
        INSERT INTO user (email, firstname, lastname, phonenumber, address,
            api_token, code_role, password_hash)
        VALUES(:EMAIL, :FIRSTNAME, :LASTNAME, :PHONENUMBER, :ADDRESS, :API_TOKEN, :
            CODE_ROLE, :PASSWORD_HASH)";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':EMAIL', $user->email, \PDO::PARAM_STR);
            $statement->bindParam(':FIRSTNAME', $user->firstname, \PDO::PARAM_STR);
            $statement->bindParam(':LASTNAME', $user->lastname, \PDO::PARAM_STR);
            $statement->bindParam(':PHONENUMBER', $user->phonenumber, \PDO::
                PARAM_STR);
            $statement->bindParam(':ADDRESS', $user->address, \PDO::PARAM_STR);
            $statement->bindParam(':API_TOKEN', $user->api_token, \PDO::PARAM_STR);
            $statement->bindParam(':CODE_ROLE', $user->code_role, \PDO::PARAM_INT);
            $statement->bindParam(':PASSWORD_HASH', $user->password_hash, \PDO::
                PARAM_STR);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a user in the database.
     *
     * @param User $user The user model object
     * @return int The number of rows affected by the update
     */
    public function update(User $user)
    {
        $statement = "
        UPDATE user
        SET email = :EMAIL,
        firstname = :FIRSTNAME,
        lastname = :LASTNAME,
        phonenumber = :PHONENUMBER,
        address = :ADDRESS,
        api_token = :API_TOKEN,
        code_role = :CODE_ROLE,
        password_hash = :PASSWORD_HASH
        WHERE id = :ID_USER";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':EMAIL', $user->email, \PDO::PARAM_STR);
            $statement->bindParam(':FIRSTNAME', $user->firstname, \PDO::PARAM_STR);
            $statement->bindParam(':LASTNAME', $user->lastname, \PDO::PARAM_STR);
            $statement->bindParam(':PHONENUMBER', $user->phonenumber, \PDO::
                PARAM_STR);
            $statement->bindParam(':ADDRESS', $user->address, \PDO::PARAM_STR);
            $statement->bindParam(':API_TOKEN', $user->api_token, \PDO::PARAM_STR);
            $statement->bindParam(':CODE_ROLE', $user->code_role, \PDO::PARAM_STR);
            $statement->bindParam(':PASSWORD_HASH', $user->password_hash, \PDO::
                PARAM_STR);
            $statement->bindParam(':ID_USER', $user->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a user in the database.
     *
     * @param User $user The user model object
     * @return int The number of rows affected by the delete
     */
    public function delete(User $user)
    {
        $statement = "
        DELETE FROM user
        WHERE id = :ID_USER";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_USER', $user->id, \PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a user from the database user from his api token.
     *
     * @param string $api_token The user api token
     * @return User A User model object containing all the result rows of the query
     */
    public function findByApiToken(string $api_token)
    {
        $statement = "
        SELECT id, email, firstname, lastname, phonenumber, address, api_token,
            code_role,password_hash
        FROM user
        WHERE api_token = :API_TOKEN
        LIMIT 1";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':API_TOKEN', $api_token, \PDO::PARAM_STR);
            $statement->execute();

            $user = new User();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $user->id = $result["id"];
                $user->email = $result["email"];
                $user->firstname = $result["firstname"];
                $user->lastname = $result["lastname"];
                $user->phonenumber = $result["phonenumber"];
                $user->address = $result["address"];
                $user->api_token = $result["api_token"];
                $user->code_role = $result["code_role"];
            }
            else{
                $user = null;
            }

            return $user;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return a user from the database user from his email.
     *
     * @param string $email The user email
     * @return User A User model object containing all the result rows of the query
     */
    public function findUserByEmail(string $email)
    {
```

```php
        $statement = "
        SELECT id, email, firstname, lastname, phonenumber, address, api_token,
           code_role, password_hash
        FROM user
        WHERE email = :EMAIL
        LIMIT 1";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':EMAIL', $email, \PDO::PARAM_STR);
            $statement->execute();

            $user = new User();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $user->id = $result["id"];
                $user->email = $result["email"];
                $user->firstname = $result["firstname"];
                $user->lastname = $result["lastname"];
                $user->phonenumber = $result["phonenumber"];
                $user->address = $result["address"];
                $user->api_token = $result["api_token"];
                $user->code_role = $result["code_role"];
                $user->password_hash = $result["password_hash"];
            }
            else {
                $user = null;
            }

            return $user;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 18 – ./Sources/app/DataAccessObject/DAOWeeklySchedule.php

```php
<?php
/**
 * DAOWeeklySchedule.php
 *
 * Data access object of the weekly_schedule table.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\DataAccessObject;

use App\Models\WeeklySchedule;

class DAOWeeklySchedule {

    private $db = null;

    /**
     *
     * Constructor of the DAOWeeklySchedule object.
     *
     * @param PDO $db The database connection
     */
    public function __construct(\PDO $db)
    {
        $this->db = $db;
    }

    /**
     *
     * Method to return all the weekly schedule of the database in an array of
         weeklyschedule objects.
     *
     * @param bool $isDeleted  Bool to define whether to search for existing or
         deleted weeklyschedules
     * @param int $idEducator The educator identifier
     * @return WeeklySchedule[] A WeeklySchedule object array
     */
    public function findAll(bool $deleted, int $idEducator)
    {
        $statement = "
        SELECT id, date_valid_from, date_valid_to, id_educator
        FROM weekly_schedule
        WHERE is_deleted= :DELETED
        AND id_educator = :ID_EDUCATOR
        ORDER BY date_valid_from";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
            $statement->execute();
            $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

            $weeklyScheduleArray = array();

            foreach ($results as $result) {
                $weeklySchedule = new WeeklySchedule();
                $weeklySchedule->id = $result["id"];
                $weeklySchedule->date_valid_from = $result["date_valid_from"];
                $weeklySchedule->date_valid_to = $result["date_valid_to"];
                $weeklySchedule->id_educator = $result["id_educator"];
                array_push($weeklyScheduleArray,$weeklySchedule);
            }

            return $weeklyScheduleArray;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to return an weekly schedule from the database in a weeklyschedule
         model object.
     *
     * @param int $id The weeklyschedule identifier
     * @param int $idEducator The educator identifier
     * @return WeeklySchedule A WeeklySchedule model object containing all the
         result rows of the query
     */
    public function find(int $id, int $idEducator)
    {
        $statement = "
        SELECT id, date_valid_from, date_valid_to, id_educator
        FROM weekly_schedule
        WHERE id = :ID_WEEKLY_SCHEDULE
        AND is_deleted = 0
        AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $id, \PDO::PARAM_INT);
            $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
            $statement->execute();

            $weeklySchedule = new WeeklySchedule();

            if ($statement->rowCount()==1) {
                $result = $statement->fetch(\PDO::FETCH_ASSOC);
                $weeklySchedule->id = $result["id"];
                $weeklySchedule->date_valid_from = $result["date_valid_from"];
                $weeklySchedule->date_valid_to = $result["date_valid_to"];
                $weeklySchedule->id_educator = $result["id_educator"];
            }
            else{
                $weeklySchedule = null;
            }

            return $weeklySchedule;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to insert a weekly schedule in the database.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return int The number of rows affected by the insert
     */
    public function insert(WeeklySchedule $weeklySchedule)
    {
        $statement = "
        INSERT INTO weekly_schedule (date_valid_from, date_valid_to,id_educator,
            is_deleted)
        VALUES(STR_TO_DATE(:DATE_VALID_FROM, \"%Y-%m-%d\"),STR_TO_DATE(:
            DATE_VALID_TO, \"%Y-%m-%d\"),:ID_EDUCATOR, 0)";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE_VALID_FROM', $weeklySchedule->
                date_valid_from, \PDO::PARAM_STR);
            $statement->bindParam(':DATE_VALID_TO', $weeklySchedule->date_valid_to,
                 \PDO::PARAM_STR);
            $statement->bindParam(':ID_EDUCATOR', $weeklySchedule->id_educator, \
                PDO::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to update a weekly schedule in the database.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return int The number of rows affected by the update
     */
    public function update(WeeklySchedule $weeklySchedule)
    {
        $statement = "
        UPDATE weekly_schedule
        SET date_valid_from = STR_TO_DATE(:DATE_VALID_FROM, \"%Y-%m-%d\"),
        date_valid_to = STR_TO_DATE(:DATE_VALID_TO, \"%Y-%m-%d\")
        WHERE id = :ID_WEEKLY_SCHEDULE";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE_VALID_FROM', $weeklySchedule->
                date_valid_from, \PDO::PARAM_STR);
            $statement->bindParam(':DATE_VALID_TO', $weeklySchedule->date_valid_to,
                 \PDO::PARAM_STR);
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $weeklySchedule->id, \PDO
                ::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to delete a weekly schedule in the database.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return int The number of rows affected by the delete
     */
    public function delete(WeeklySchedule $weeklySchedule)
    {
        $statement = "
        UPDATE weekly_schedule
        SET is_deleted = 1
        WHERE id = :ID_WEEKLY_SCHEDULE";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_WEEKLY_SCHEDULE', $weeklySchedule->id, \PDO
                ::PARAM_INT);
            $statement->execute();
            return $statement->rowCount();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to check if a start date and an end date of a weekly schedule do not
         cause an overlapping problem.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return array The associative array containing all the result rows of the
         query
     */
    public function findOverlap(WeeklySchedule $weeklySchedule)
    {
        $statement = "
        SELECT *
        FROM weekly_schedule
        WHERE (STR_TO_DATE(:DATE_VALID_FROM, \"%Y-%m-%d\") < date_valid_to OR
            date_valid_to IS NULL)
        AND (STR_TO_DATE(:DATE_VALID_TO, \"%Y-%m-%d\") > date_valid_from
        OR (STR_TO_DATE(:DATE_VALID_TO, \"%Y-%m-%d\") IS NULL AND (STR_TO_DATE(:
            DATE_VALID_FROM, \"%Y-%m-%d\") < date_valid_to))
        AND id_educator = :ID_EDUCATOR
        AND is_deleted = 0";

        if (!isset($input['date_valid_to'])) {
            $input['date_valid_to'] = null;
        }

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':DATE_VALID_FROM', $weeklySchedule->
                date_valid_from, \PDO::PARAM_STR);
            $statement->bindParam(':DATE_VALID_TO', $weeklySchedule->date_valid_to,
                 \PDO::PARAM_STR);
            $statement->bindParam(':ID_EDUCATOR', $weeklySchedule->id_educator, \
                PDO::PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method to check if the database contains a weekly schedule not deleted
         without end date.
     *
     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
     * @return array The associative array containing all the result rows of the
         query
     */
    public function findActifPermanentSchedule(WeeklySchedule $weeklySchedule)
    {
        $statement = "
        SELECT *
        FROM weekly_schedule
        WHERE date_valid_to IS NULL
        AND is_deleted = 0
        AND id != :ID_HIMSELF
                AND id_educator = :ID_EDUCATOR";

        try {
            $statement = $this->db->prepare($statement);
            $statement->bindParam(':ID_EDUCATOR', $weeklySchedule->id_educator, \
                PDO::PARAM_INT);
            $statement->bindParam(':ID_HIMSELF', $weeklySchedule->id, \PDO::
                PARAM_INT);
            $statement->execute();
            $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
            return $result;
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }
}
```

Listing 19 – ./Sources/app/Models/Absence.php

```
<?php
/**
 * Absence.php
 *
 * Absence model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class Absence {

    public ?int $id;
    public ?string $date_absence_from;
    public ?string $date_absence_to;
    public ?string $description;
    public ?int $id_educator;

    /**
     *
     * Constructor of the Absence model object.
     *
     * @param int $id The Absence identifier
     * @param string $date_absence_from The start date of the absences
     * @param string $date_absence_to The end date of the absences
     * @param string $description The description of the absences
     * @param int $id_educator The educator identifier
     */
    public function __construct(int $id = null, string $date_absence_from = null,
        string $date_absence_to = null,string $description = null,int $id_educator
        = null)
    {
        $this->id = $id;
        $this->date_absence_from = $date_absence_from;
        $this->date_absence_to = $date_absence_to;
        $this->description = $description;
        $this->id_educator = $id_educator;
    }
}
```

Listing 20 – ./Sources/app/Models/Appointment.php

```php
<?php
/**
 * Appointment.php
 *
 * Appointment model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class Appointment {

    public ?int $id;
    public ?string $datetime_appointment;
    public ?int $duration_in_hour;
    public ?string $note_text;
    public ?string $note_graphical_serial_id;
    public ?string $summary;
    public ?int $user_id_customer;
    public ?int $user_id_educator;

    /**
     *
     * Constructor of the Appointment model object.
     *
     * @param int $id The Appointment identifier
     * @param string $datetime_appointment The date of the appointment
     * @param int $duration_in_hour The duration of the appointment in hours
     * @param string $note_text The text note of the appointment
     * @param string $note_graphical_serial_id The serial id of the graphical note
     * @param string $summary The educator identifier
     * @param int $user_id_customer The customer identifier
     * @param int $user_id_educator The educator identifier
     */
    public function __construct(int $id = null, string $datetime_appointment = null
        , int $duration_in_hour = null,string $note_text = null,string
        $note_graphical_serial_id = null,string $summary = null, int
        $user_id_customer = null, int $user_id_educator = null)
    {
        $this->id = $id;
        $this->datetime_appointment = $datetime_appointment;
        $this->duration_in_hour = $duration_in_hour;
        $this->note_text = $note_text;
        $this->note_graphical_serial_id = $note_graphical_serial_id;
        $this->summary = $summary;
        $this->user_id_customer = $user_id_customer;
        $this->user_id_educator = $user_id_educator;
    }
}
```

Listing 21 – ./Sources/app/Models/Document.php

```php
<?php
/**
 * Document.php
 *
 * Document model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class Document {

    public ?int $id;
    public ?string $document_serial_id;
    public ?string $type;
    public ?int $user_id;
    public ?int $package_number;
    public ?string $signature_base64;

    /**
     *
     * Constructor of the Document model object.
     *
     * @param int $id The document identifier
     * @param string $document_serial_id The serial id of the document
     * @param string $type The type of the document
     * @param int $user_id The identifier of the owner of the document
     * @param int $package_number The package number (Number from 1 to 5)
     * @param string $signature_base64 The image of the signature in base64
     */
    public function __construct(int $id = null, string $document_serial_id = null,
        string $type = null,int $user_id = null,int $package_number = null, string
        $signature_base64 = null)
    {
        $this->id = $id;
        $this->document_serial_id = $document_serial_id;
        $this->type = $type;
        $this->user_id = $user_id;
        $this->package_number = $package_number;
        $this->signature_base64 = $signature_base64;
    }
}
```

Listing 22 – ./Sources/app/Models/Dog.php

```php
<?php
/**
 * Dog.php
 *
 * Dog model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class Dog {

    public ?int $id;
    public ?string $name;
    public ?string $breed;
    public ?string $sex;
    public ?string $picture_serial_id;
    public ?string $chip_id;
    public ?int $user_id;

    /**
     *
     * Constructor of the Dog model object.
     *
     * @param int $id The dog identifier
     * @param string $name The name of the dog
     * @param string $breed The breed of the dog
     * @param string $sex The sex of the dog
     * @param string $picture_serial_id The picture serial id of the dog
     * @param string $chip_id The chip id of the dog
     * @param int $user_id The user id of the dog's owner
     */
    public function __construct(int $id = null, string $name = null, string $breed
        = null,
     string $sex = null, string $picture_serial_id = null, string $chip_id = null,
        int $user_id = null)
    {
        $this->id = $id;
        $this->name = $name;
        $this->breed = $breed;
        $this->sex = $sex;
        $this->picture_serial_id = $picture_serial_id;
        $this->chip_id = $chip_id;
        $this->user_id = $user_id;
    }
}
```

Listing 23 – ./Sources/app/Models/ScheduleOverride.php

```php
<?php
/**
 * ScheduleOverride.php
 *
 * ScheduleOverride model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class ScheduleOverride {

    public ?int $id;
    public ?string $date_schedule_override;
    public ?int $id_educator;

    /**
     *
     * Constructor of the ScheduleOverride model object.
     *
     * @param int $id The ScheduleOverride identifier
     * @param string $date_valid_from The date of the schedule override
     * @param int $id_educator The educator identifier
     */
    public function __construct(int $id = null, string $date_schedule_override =
        null ,int $id_educator = null)
    {
        $this->id = $id;
        $this->date_schedule_override = $date_schedule_override;
        $this->id_educator = $id_educator;
    }
}
```

Listing 24 – ./Sources/app/Models/TimeSlot.php

```
<?php
/**
 * TimeSlot.php
 *
 * TimeSlot model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class TimeSlot {

    public ?int $id;
    public ?int $code_day;
    public ?string $time_start;
    public ?string $time_end;
    public ?int $id_weekly_schedule;
    public ?int $id_schedule_override;
    public ?int $id_educator;

    /**
     *
     * Constructor of the TimeSlot model object.
     *
     * @param int $id The TimeSlot identifier
     * @param int $code_day The day code of a week (1 = Sunday, 2 = Monday, 3 =
         Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, 7 = Saturday)
     * @param string $time_start The start time of the time slot
     * @param string $time_end The end time of the time slot
     * @param int $id_weekly_schedule The weekly schedule identifier
     * @param int $id_schedule_override The schedule override identifier
     * @param int $id_educator The educator identifier
     */
    public function __construct(int $id = null, int $code_day = null, string
        $time_start = null, string $time_end = null, int $id_weekly_schedule = null
        , int $id_schedule_override = null, int $id_educator = null)
    {
        $this->id = $id;
        $this->code_day = $code_day;
        $this->time_start = $time_start;
        $this->time_end = $time_end;
        $this->id_weekly_schedule = $id_weekly_schedule;
        $this->id_schedule_override = $id_schedule_override;
        $this->id_educator = $id_educator;
    }
}
```

Listing 25 – ./Sources/app/Models/User.php

```
<?php
/**
```

```
 * User.php
 *
 * User model.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class User {

    public ?int $id;
    public ?string $email;
    public ?string $firstname;
    public ?string $lastname;
    public ?string $phonenumber;
    public ?string $address;
    public ?string $api_token;
    public ?int $code_role;
    public ?string $password_hash;

    /**
     *
     * Constructor of the User model object.
     *
     * @param int $id The user identifier
     * @param string $email The email of the user
     * @param string $firstname the first name of the user
     * @param string $lastname the last name of the user
     * @param string $phonenumber The phone number of the user
     * @param string $address The address of the user
     * @param string $api_token The api_token of the user
     * @param int $code_role The code_role of the user
     * @param string $password_hash The password_hash of the user
     */
    public function __construct(int $id = null, string $email = null, string
        $firstname = null,
     string $lastname = null, string $phonenumber = null, string $address = null,
         string $api_token = null, int $code_role = null, string $password_hash =
         null)
    {
        $this->id = $id;
        $this->email = $email;
        $this->firstname = $firstname;
        $this->lastname = $lastname;
        $this->phonenumber = $phonenumber;
        $this->address = $address;
        $this->api_token = $api_token;
        $this->code_role = $code_role;
        $this->password_hash = $password_hash;
    }
}
```

Listing 26 – ./Sources/app/Models/WeeklySchedule.php

```
<?php
/**
 * WeeklySchedule.php
 *
 * WeeklySchedule model.
```

```php
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\Models;

class WeeklySchedule {

    public ?int $id;
    public ?string $date_valid_from;
    public ?string $date_valid_to;
    public ?int $id_educator;

    /**
     *
     * Constructor of the WeeklySchedule model object.
     *
     * @param int $id The WeeklySchedule identifier
     * @param string $date_valid_from The start date of the weekly schedule
     * @param string $date_valid_to The end date of the weekly schedule
     * @param int $id_educator The educator identifier
     */
    public function __construct(int $id = null, string $date_valid_from = null,
        string $date_valid_to = null,int $id_educator = null)
    {
        $this->id = $id;
        $this->date_valid_from = $date_valid_from;
        $this->date_valid_to = $date_valid_to;
        $this->id_educator = $id_educator;
    }
}
```

Listing 27 – ./Sources/app/System/Constants.php

```php
<?php
/**
 * Constants.php
 *
 * Class allowing the use of the constants of the application.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\System;

class Constants
{
    const USER_CODE_ROLE = 1;
    const ADMIN_CODE_ROLE = 2;
    const DOCUMENT_TYPE_CONDTIONS_OF_REGISTRATION = "conditions_inscription";
    const DOCUMENT_TYPE_POSTER = "poster";
    const IMAGE_TYPE_JPEG = "image/jpeg";
    const IMAGE_TYPE_PNG = "image/png";
    const TYPE_DOCUMENT_PDF = "application/pdf";
}
```

Listing 28 – ./Sources/app/System/DatabaseConnector.php

```php
<?php
/**
 * DatabaseConnector.php
 *
 * Class allowing the connection to the database with the environment variables of
     the .env file.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
namespace App\System;

class DatabaseConnector {

    private \PDO $dbConnection;

    /**
     *
     * Constructor of the User object.
     *
     */
    public function __construct()
    {
        $host = getenv('DB_HOST');
        $port = getenv('DB_PORT');
        $db   = getenv('DB_DATABASE');
        $user = getenv('DB_USERNAME');
        $password = getenv('DB_PASSWORD');

        try {
            $this->dbConnection = new \PDO("mysql:host=$host;port=$port;charset=
                utf8mb4;dbname=$db", $user, $password);
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    }

    /**
     *
     * Method for returning the connection.
     *
     * @return PDO The database connection
     */
    public function getConnection()
    {
        return $this->dbConnection;
    }
}
```

Listing 29 – ./Sources/bootstrap.php

```php
<?php
/**
 * bootstrap.php
 *
 * Bootable file of the API.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

require 'vendor/autoload.php';
use Dotenv\Dotenv;
use App\System\DatabaseConnector;
```

```

// Loads the environment variables from the .env file
$dotenv = new DotEnv(__DIR__);
$dotenv->load();

$dbConnection = (new DatabaseConnector())->getConnection();
```

Listing 30 – ./Sources/public/v1/absences/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process absence
    requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\AbsenceController;
use App\Models\Absence;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
  header('Allow: GET,POST,PATCH,DELETE');
  return;
}

$controller = new AbsenceController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$absence = new Absence();
$absence->id = $id ?? null;
$absence->date_absence_from = $input["date_absence_from"] ?? null;
$absence->date_absence_to = $input["date_absence_to"] ?? null;
$absence->description = $input["description"] ?? null;

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllAbsences();
        }
        else{
            $response = $controller->getAbsence($id);
        }
        break;

    case 'POST':
        $response = $controller->createAbsence($absence);
        break;

    case 'PATCH':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->updateAbsence($absence);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteAbsence($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 31 – ./Sources/public/v1/appointments/downloadNoteGraphical/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process download note
      graphical.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\AppointmentController;

require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

```

```php
if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET,POST,PATCH,DELETE');
        return;
}

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$serial_id = end($pathFragments);

$controller = new AppointmentController($dbConnection);

switch ($requestMethod) {
    case 'GET':
        if (empty($serial_id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->downloadNoteGraphical($serial_id);
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 32 – ./Sources/public/v1/appointments/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process appointment
    requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\AppointmentController;
use App\Models\Appointment;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET,POST,PATCH,DELETE');
        return;
```

```php
}

$controller = new AppointmentController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$appointment = new Appointment();
$appointment->id = $id ?? null;
$appointment->datetime_appointment = $input["datetime_appointment"] ?? null;
$appointment->duration_in_hour = isset($input["duration_in_hour"]) && is_numeric(
    $input["duration_in_hour"]) ? $input["duration_in_hour"] : null;
$appointment->note_text = $input["note_text"] ?? null;
$appointment->note_graphical_serial_id = $input["note_graphical_serial_id"] ?? null
    ;
$appointment->summary = $input["summary"] ?? null;
$appointment->user_id_customer = isset($input["user_id_customer"]) && is_numeric(
    $input["user_id_customer"]) ? $input["user_id_customer"] : null;
$appointment->user_id_educator = isset($input["user_id_educator"]) && is_numeric(
    $input["user_id_educator"]) ? $input["user_id_educator"] : null;


switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllAppointments();
        }
        else{
            $response = $controller->getAppointment($id);
        }
        break;

    case 'POST':
        $response = $controller->createAppointment($appointment);
        break;

    case 'PATCH':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->updateAppointment($appointment);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteAppointment($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
```

```
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 33 – ./Sources/public/v1/appointments/uploadNoteGraphical/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process upload note
    graphical.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\AppointmentController;

require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET,POST,PATCH,DELETE');
        return;
}

$controller = new AppointmentController($dbConnection);

switch ($requestMethod) {
    case 'POST':
        $response = $controller->uploadNoteGraphical();
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 34 – ./Sources/public/v1/connection/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process connection
     request.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\UserController;
use App\Models\User;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: POST');
        return;
}

$controller = new UserController($dbConnection);

parse_str(file_get_contents('php://input'), $input);

$user = new User();
$user->email = $input["email"] ?? null;
$user->password_hash = $input["password"] ?? null;

switch ($requestMethod) {
    case 'POST':
        $response = $controller->connection($user);
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 35 – ./Sources/public/v1/documents/downloadDocument/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process download
     document.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
```

```php
 */

use App\Controllers\DocumentController;

require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET');
        return;
}

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$serial_id = end($pathFragments);

$controller = new DocumentController($dbConnection);

switch ($requestMethod) {
    case 'GET':
        if (empty($serial_id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->downloadDocument($serial_id);
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 36 – ./Sources/public/v1/documents/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process document
    requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\DocumentController;
use App\Models\Document;
```

```php

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET,POST,PATCH,DELETE');
        return;
}

$controller = new DocumentController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

$document = new Document();
$document->id = $id ?? null;
$document->document_serial_id = $_POST["document_serial_id"] ?? null;
$document->type = $_POST["type"] ?? null;
$document->user_id = isset($_POST["user_id"]) && is_numeric($_POST["user_id"]) ?
    $_POST["user_id"] : null;
$document->package_number =  isset($_POST["package_number"]) && is_numeric($_POST["
    package_number"]) ? $_POST["package_number"] : null;
$document->signature_base64 = $_POST["signature_base64"] ?? null;

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllDocuments();
        }
        else{
            $response = $controller->getDocument($id);
        }
        break;

    case 'POST':
        $response = $controller->createDocument($document);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteDocument($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 37 – ./Sources/public/v1/dogs/downloadPicture/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process download dog
    picture.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\DogController;

require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$serial_id = end($pathFragments);

$controller = new DogController($dbConnection);

switch ($requestMethod) {
    case 'GET':
        if (empty($serial_id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->downloadDogPicture($serial_id);
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 38 – ./Sources/public/v1/dogs/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process dog requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\DogController;
use App\Models\Dog;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET,POST,PATCH,DELETE');
        return;
}

$controller = new DogController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$dog = new Dog();
$dog->id = $id ?? null;
$dog->name = $input["name"] ?? null;
$dog->breed = $input["breed"] ?? null;
$dog->sex = $input["sex"] ?? null;
$dog->chip_id = $input["chip_id"] ?? null;
$dog->user_id = isset($input["user_id"]) && is_numeric($input["user_id"]) ? $input[
    "user_id"] : null;

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllDogs();
        }
        else{
            $response = $controller->getDog($id);
        }
        break;

    case 'POST':
        $response = $controller->createDog($dog);
        break;

    case 'PATCH':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->updateDog($dog);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteDog($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 39 – ./Sources/public/v1/dogs/uploadPicture/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process upload dog
     picture.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\DogController;

require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: POST');
        return;
}

$controller = new DogController($dbConnection);

```

```php
switch ($requestMethod) {
    case 'POST':
        $response = $controller->uploadDogPicture();
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 40 – ./Sources/public/v1/index.php

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>api-rest-douceur-de-chien</title>
</head>
<body>
    <h1>api-rest-douceur-de-chien.boreljaquet.ch</h1><p>Cette API est en cours de d
        éveloppement.</p>
</body>
</html>
```

Listing 41 – ./Sources/public/v1/plannings/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to display plannings.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\TimeSlotController;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

$controller = new TimeSlotController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->getPlanningTimeSlots($id);
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 42 – ./Sources/public/v1/scheduleOverrides/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process schedule
     override requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\ScheduleOverrideController;
use App\Models\ScheduleOverride;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
  header('Allow: GET,POST,DELETE');
  return;
}

$controller = new ScheduleOverrideController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$scheduleOverride = new ScheduleOverride();
$scheduleOverride->id = $id ?? null;
```

```
$scheduleOverride->date_schedule_override = $input["date_schedule_override"] ??
    null;

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllScheduleOverrides();
        }
        else{
            $response = $controller->getScheduleOverride($id);
        }
        break;

    case 'POST':
        $response = $controller->createScheduleOverride($scheduleOverride);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteScheduleOverride($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 43 – ./Sources/public/v1/timeSlots/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process time slot
    requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\TimeSlotController;
use App\Models\TimeSlot;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
```

```

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
  header('Allow: GET,POST,DELETE');
  return;
}

$controller = new TimeSlotController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$timeSlot = new TimeSlot();
$timeSlot->id = $id ?? null;
$timeSlot->code_day = isset($input["code_day"]) && is_numeric($input["code_day"]) ?
     $input["code_day"] : null;
$timeSlot->time_start = $input["time_start"] ?? null;
$timeSlot->time_end = $input["time_end"] ?? null;
$timeSlot->id_weekly_schedule = isset($input["id_weekly_schedule"]) && is_numeric(
    $input["id_weekly_schedule"]) ? $input["id_weekly_schedule"] : null;
$timeSlot->id_schedule_override = isset($input["id_schedule_override"]) &&
    is_numeric($input["id_schedule_override"]) ? $input["id_schedule_override"] :
    null;


switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllTimeSlots();
        }
        else{
            $response = $controller->getTimeSlot($id);
        }
        break;

    case 'POST':
        $response = $controller->createTimeSlot($timeSlot);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteTimeSlot($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
```

```php
    echo $response['body'];
}
```

Listing 44 – ./Sources/public/v1/users/educators/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process educator requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\UserController;


require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

$controller = new UserController($dbConnection);

switch ($requestMethod) {
    case 'GET':
        $response = $controller->getAllEducatorUsers();
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 45 – ./Sources/public/v1/users/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process user requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\UserController;
use App\Models\User;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE,OPTIONS");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
        header('Allow: GET,POST,PATCH,DELETE');
        return;
}

$controller = new UserController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$user = new User();
$user->id = $id ?? null;
$user->email = $input["email"] ?? null;
$user->firstname = $input["firstname"] ?? null;
$user->lastname = $input["lastname"] ?? null;
$user->phonenumber = $input["phonenumber"] ?? null;
$user->address = $input["address"] ?? null;
$user->api_token = $input["api_token"] ?? null;
$user->code_role = $input["code_role"] ?? null;
$user->password_hash = $input["password"] ?? null;
$reCAPTCHAuserResponseToken = $input["reCAPTCHAuserResponseToken"] ?? null;

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllCustomerUsers();
        }
        else{
            $response = $controller->getUser($id);
        }
        break;

    case 'POST':
        $response = $controller->createUser($user,$reCAPTCHAuserResponseToken);
        break;

    case 'PATCH':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->updateUser($user);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteUser($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 46 – ./Sources/public/v1/users/me/changePassword/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process the password
     change request current logged in user request.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\UserController;
use App\Models\User;

require "../../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: PATCH, OPTIONS");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
  header('Allow: PATCH');
  return;
}

$controller = new UserController($dbConnection);

parse_str(file_get_contents('php://input'), $input);

$user = new User();
$user->password_hash = $input["password"] ?? null;

switch ($requestMethod) {
    case 'PATCH':
        $response = $controller->updateMyPassword($user);
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 47 – ./Sources/public/v1/users/me/index.php

```
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process current
     logged in user request.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\UserController;

require "../../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET, OPTIONS");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
  header('Allow: GET');
  return;
}

$controller = new UserController($dbConnection);

switch ($requestMethod) {
    case 'GET':
        $response = $controller->getMyInformations();
        break;
    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 48 – ./Sources/public/v1/weeklySchedules/index.php

```php
<?php
/**
 * index.php
 *
 * File being the front controller of the API and allowing to process weekly
     schedule requests.
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */

use App\Controllers\WeeklyScheduleController;
use App\Models\WeeklySchedule;

require "../../../bootstrap.php";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET,POST,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

$requestMethod = $_SERVER["REQUEST_METHOD"];

if (strcmp("OPTIONS", $requestMethod) == 0) {
  header('Allow: GET,POST,DELETE');
  return;
}

$controller = new WeeklyScheduleController($dbConnection);

$path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$pathFragments = explode('/', $path);
$id = intval(end($pathFragments));

parse_str(file_get_contents('php://input'), $input);

$weeklySchedule = new WeeklySchedule();
$weeklySchedule->id = $id ?? null;
$weeklySchedule->date_valid_from = $input["date_valid_from"] ?? null;
$weeklySchedule->date_valid_to = $input["date_valid_to"] ?? null;

switch ($requestMethod) {
    case 'GET':
        if (empty($id) || !is_numeric($id)) {
            $response = $controller->getAllWeeklySchedules();
        }
        else{
            $response = $controller->getWeeklySchedule($id);
        }
        break;

    case 'POST':
        $response = $controller->createWeeklySchedule($weeklySchedule);
        break;

    case 'DELETE':
        if (empty($id) || !is_numeric($id)) {
            header("HTTP/1.1 404 Not Found");
            exit();
        }
        $response = $controller->deleteWeeklySchedule($id);
        break;

    default:
        header("HTTP/1.1 404 Not Found");
        exit();
        break;
}

header($response['status_code_header']);
if ($response['body']) {
    echo $response['body'];
}
```

Listing 49 – ./Sources/resources/template/conditions//textunderscoreregistration.php

```
<!--
    conditions_registration.php

    Template of the registration conditions content used in the HelperController
       with the storeConditionsRegistration function.

    @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 -->

<!DOCTYPE html>
<html>

<head>
    <title>condition_inscription.pdf</title>
</head>
<style>
textarea#mentions {
  height: 350px;
}
strong{
        color: #3598db;
}
figcaption{
    margin-left: 20px;
}

div.card,
.tox div.card {
  width: 240px;
  background: white;
  border: 1px solid #ccc;
  border-radius: 3px;
  box-shadow: 0 4px 8px 0 rgba(34, 47, 62, .1);
  padding: 8px;
  font-size: 14px;
  font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,Oxygen-Sans,
      Ubuntu,Cantarell,"Helvetica Neue",sans-serif;
}

div.card::after,
.tox div.card::after {
  content: "";
  clear: both;
  display: table;
}

div.card h1,
.tox div.card h1 {
  font-size: 14px;
  font-weight: bold;
  margin: 0 0 8px;
  padding: 0;
  line-height: normal;
  font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,Oxygen-Sans,
      Ubuntu,Cantarell,"Helvetica Neue",sans-serif;
}

div.card p,
.tox div.card p {
  font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,Oxygen-Sans,
      Ubuntu,Cantarell,"Helvetica Neue",sans-serif;
}

div.card img.avatar,
.tox div.card img.avatar {
  width: 48px;
  height: 48px;
  margin-right: 8px;
  float: left;
}
</style>

<body>
    <p style="text-align: center; font-size: 15px;"><img title="TinyMCE Logo"
        src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAA+
            gAAAQmCAMAAACUOGWcAAAAkFBMVEUAAAA+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+
            o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g
            +o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+o9g+
            o9g+o9g+o9isQD+DAAAAL3RSTlMAAwX27wkP4tvo1BQZNc7IW7hiKqxJIS+
            CVaYdmMJ7vYdtRCY7nXKyT42SP3ZpoqyufvAAALL1SURBVHja7NrpcqJAFIbh06cRUNyQTRPBfY
            +e+7+7mZAYGyLozCRVM+P3pJKyiijJj9fTjRAAAAAAAAAAAAAAAAAAAMB/QREA/
            I94kQRBsIg9Vqgd4B+ich+P60d4O9VOO3Fb4TBbR53kmQkA/maqWDXbbfvWjGaaO3JhNSfT
            /mFBRApzHeAvpNRH3otkGY0GL9NVuMoG0TLmutQ3oiWnf5Jcd/
            RHc13hTQLgmygi7mz7x2x32vRcSy6syW7LdF3jkLlSlNfuv8SBp5juppjJxGgd4Du0t3u30KsWnf98NV7SFY
            +mcSP3axfqG5zXyFUWA1gG+
            gz3b6PPyW0uJFnHWV6plmomWKuEsIqZ7MFEymg33qzA87cZPftrb9wNcvgf4cp2850papH+
            t2iepk9JdmBpZS0p6u0GHFC7hA3ylnWipZz1TiaKlK9W0uNmc1B3L9vjpbW9//
            pL8W1ozIpQO8GXssSU3TTqfEvV8qeNaqUd8s/OoVbGIkPGWGgQAX2Mk9/
            DL45lpLzX0ZiLrm0OZj5boqv2C28BMB/gNSjErKjnKPXSsqORUu+
            J3LKeftEnVDvRt7cUBf45LcgC/eVcMU1Ff7rIqP7GxKc/
            gksnaz7zazgNHdN1LtJYoHeDXMJG36B8/bZwHchfrmUv3uU/E0HU/
            hapdS29JVXdub0QXM28Zr+E0RUKyqQYz4wN3gKJk3O02de9AtaHrnz4eimFPXBzHXTFMk/5
            GSvSquyamKioTk5buYGFcm7OOSDMgvnWrDWMjDw9I8Sv6xNs5krMy++
            ZEt9w0da3ijLY8rvl4bU3UXg5LS3FHVg2qNC91niZEFO/E5M6Iqztf7/
            ezmAAejuLLftykKLHkbESqKnTrKZuNDp1k8fy8SDrrphimZnNMkXFQy/r1GCd+
            ofRWaG0rQ41dKZgE+Z/
```

diJzzRNf5OqJBVZ60iHb9o0cAD0URBYPh2B8rUlQUiT5HGXqVE33CZlcc74xum7YyQ+9
rMaxVfvb2WMxnuBLaFaXzS2mDPj//
B4ePPf5GJA2qdvlqef69KT6Gg4eiKD41LS0iIZePREZXEamK0EPi4h64ZbRorgSYjlIKPWdnosU0peulxo4Y
+Y1S6B3rEtXYq5zoxayYMn2JcUhsHhFTn97ZPTFZs9GCrhmUtvPJ5czckjerg+
hefL10FV3OMcBIhweiKLgM4NSmgjiViw6p66Gn5ZWAbezEJ4XQh2Lq/
mDvTJsUx2EwLNu5IQnh6nCEhuY+9f//3W4tC7LsOHvVznbN+
vk0NQISungtWZKVUN8jENn99gEt9DPu0Dctpf1gdsMqaVex0GONoxe6538Ez4TvgBHpu2d1IsOsS
+gS9mRMpduj014hqjR/HqebB9so0FUViyQW+
otk8nz3OGniZmWq2L7tJDcvIfykGs9Piz7DTeEQOBtUZJzBm4kVuvMgISDZfoFwCF2/2
lW7UBLEM2ih94mMGb/uk04Ul21CF5AgkdEyoeGbaTw/
JxJOKb6phTMbp1CTbM6ELqzPbFqzcRIOLPSewJtozCyDWQ4ELQb6ax4RMGTxvqE6ARvBUoF26L64X1c78Hh
+SiTMAhZM82xcSrZUK1r1USPbgrA0qbmFTjy0q+1QacF7XIdA0FqgkVqnV363B+
Vx0jreRsZIKKP/TkKFClV6nIHH8/MhYRiw9Bhj8Uk2tSKjQI3YnAAlYKPahX439+
gk44d2pTjIb19g8MEd+lKCweFlK/Znu5Iu4IYayQd7hYTzyzICj+
enQ8Jd148h9PCCZI23ZEiRUBtL6JM/
JfShMzDHOlsJYIR7nnKvgeCJg897cm7pl98GqNGYDr3gKQeOjHyK3vNNEUL0vwZb93BU05xW2APOXd
+kX+
FNZh9dIYQmKjUAwTvjHBuFRUmG5JgmOADGFBlZbulx9LQEda0yKxsnwgyN3lze7RtrST5GL5ocC4yv
/vSr5/sR3Y9l/NzLqpmQ4IJnwhWGpmvWjQcBLyqjNc6Zdg8mbqHPJY+838ZicETM+
X2ukVGDRfS0BMcCq/7C/JpH1Il3IOwNjO3SP+rZHhWiwrHvjvd8N+
QpYadG01OH0Ne6liOzNy5BQvutszi66YFwqSoVwkj8EXsm9FyRRx9dS9xwv0pWqoJzwlTrrYPQmEujsGNtkr
+75TgiZfyJH4YFtMs0sFBEBp18hy7u/WHeG0TNSVQnSKfSHZFKmMVPpIy3LeZ+
fT2VKPfZt2cmG7OspG18hjTnT6dVcmhomdDvKULj33fGeb0T/8B65Qk5dYTEWjs640Q/
d0fKsb9KH7UPjghnzj8x7PkAax1SJRrob39LVesrvg5HsWoS+1/
ODDBGmqFOBNJIKSZvQ5VrvDPS7dM+3oTeMrTFqxdON5otWnS+
ajmSc4Nm4E2kZdXjH6UwxExN6jBoXYyoF76YZbIEIM2Qs06fPxPXX7otvwRVqBOTuWahhCj3a69
/+7DPvnu+BnBQkcyabABWrELu0N4M/
l43LkeAdp1xTgV6uFnBlF4tD0BllztobLIxza18gbaFvXuZsM2+
GRjeMTtzr2r80Uh06RpCDx/MN6K0dMo/
VuFEqLiNHdY2ohDlkJk0i6FNrnM7xPZ2m97Fmlhs/q35NUOOTbGbPTIDLXiics6vGEdiI+
mVOl+M7W2D6KeokIAHcC8GkrbqIClc+dPd8A8IKFbo4rfEczEG0PPaQ0QNO/6gL+
m2VqFMunkqdDirsSG4LyD03Rxe0SQ+Ce11MrqRmOecr1x5aEOe30B+
XnA28uCnUKcCcTluw2bVsN0GkvmnO8w2YoWJtpKgz3yWqsktEEgb8dYOuh6wF07c2UqbYWf
/j4zo7B0ZEkUQSOoQ+Nqwnevdona1G1hlU2i6DBAu5fN9ns1wwoTdo5+Kcs+
y0YGBa8toFeDz/NVHCzmx8D0t5E9PG81whViqy9+hGPtqamqpHr9v3/zPFqiRRz3/YRSy+
EShQY2VaE3LZ+3Q16NNtKnTvCGy3n+zrkH1w6nywhH18tgEBoKfdicSPj/X859x5h+
dv5POAdr2oTsuBY+6
EK3aXbHxT9paeqJCg2e6MYGB3zDGh19I55yJJjoMx7ZV7yBm2Cv34NKZFdlkzh75GhqpNoY
+RyISwdhNUX7Txz2X3/
Eh6KRLFFOSvhACnTPNV8W45BFt7jIFhP5FJUVuq2KMLt9sVsC1Z6B66hK7wAR8ZCT1HszwmAVwNM9k
+W80hsg+sUHDOUwd8Zl0Kgix5jDpZ3/
HUKuHF7vkx3NrqXSKEUUXyL5Na5T0jD9UgYwXcfk2RWIasNa6TREh74GzjrqBRI5pSiAPoaWn3mbtdxp48UV
/cZs+Rzz5MCszZlHu9Vl5EfQeX4IRXvQHEJv+
faG92OlJlMQRq6eseyqr6kpDZn6AxT5Y9ez1y4SnNm4+CO/7/
gqRhTsuLmZsQvi4lKBYH3sjNKM3A8Kidpee4iyBybbwSNALHpe6Z4fgFQOjycB6reIb6ju122fqWtsqkBwJ6
+
GUtcZ1sQELAb2qLRlHrheJ6ojz7ct0QcYjBMI61BIUzeUMIf0BSmSotSl0tg85td4SzdPijOryGYN8Tn307v
+pFthj4yAjBS82r/zHLupQbQIParcouGTqxTW92omfrccUSelLbpOqF6HXA/5
kPexMwKzaV2yOGPBUxglMj4j3fqsv6nnyBvweP42Av4cQ+aV+/
zo1jZFIoHdHSQZL8jpA0OwByadw3dPahfNGVqJmGKXEqz6GtEs1RcV85yzq4joZR6GuztsBQ
/MiZi92+zXMyqQ8mI8BWqnP7LuWpE19udYPf9A5BKkFH8keAHLjmNhUt/
jqtXtXOtCr5EzMT65RqWJpA9PQlToItssoB05dhxTpXwAUR7emuPXUoNWoe/eB996p6w+8/
ZWe/6r4y+wNNr1hqhaB91JgI91ghpx7pXu+Vs6jyAcPJ2VBClBdLw0Mzq/3
POXV3UJwt0DizerR1aRMXgJXQboQOHKuS6JOWrcRYdHV3gW5Ks5B5Adre7J8Dx/
z7cS1hNY1ca8uwo1Zo7xd7zEKARs6ww5mT/14
vnrSAjn43lwWZ4mX4OTfEaLLgGFqnu2Ux5ogW+yBeHsgcXH00h2JKg3TmbooPMHP3+
vGnGGe/MudykS6qoNgP0THl2UT2NyrmkxoD53vY9XdETuIyCokm5sOATAvbRXuKF36Z6/
iJAgHvgbaRwnybG+PLbuPfvU2eJpN4dV5xVI9jtnFIYOJkFby4ys0E22cAUeD9peNC+
hE4uKV7tfnDoHQ3K3ry6TS3VcPytw1AxDJF39r3h09PgQWQQC+

kuFFuXWC93z15AAJyPVjNisctc2cGGNOOUIWZCCDhVId8eMAuHskaWss9yjE4XOcTb0riCwK
+09cviIzagnedG+u4z+hU+qQzy/Qe4KWOzHNY1RYyPMexojJ4gEbJu2b70Gj+
cv6ny0D1BDZQEqVOkeRKuG+
mg9hcTZyqqWlwX0yHZEzhTcBSb1ml4h1uhG4UQ4hU7k0HUmdn582+vOMym8V0+
VcXw8iC2FMhw161rICvryLO1OKFzwgRRk+fIO/Rf2zkQ5dRyIoi3J8o6xzb7vayD6/7+
blwBpWrIcHswenZpKvYxMEihfS2p133b8FhxNGdEPKlS/iMsc+Lczet+8hkP49cyI9+88
Ib6JhD3RJlWfUOVDqXExB8SWGVcYF5WeQtCNcaoIkxEwa2qg8FRRdiBBQ1qKJ5l+
IlHvIj3VhZ4umkKZRDk4HL8DA9k2i8O866oYGLPO6PZmI+xLLd7Yy/EkqOIgfQpUCu+
V7VUG31W0IDZfd0+CPfLV7Ce02RISkowXMzIfDAfEUoLQrF25o1hJ+R4BZU6Yuio2x+/
BUHsmoi/
BJLUEzhF2c0YX7d37eAucYTCK0gZmWE8gU45dEWsQewZVokE8tYeatXuzBzahRwPgtRlDzfVIs8RD
+
rUhiDMOYqXqI0QL7nTueBScfO1kPW4XuhlvMpPao3Y03oHEko7asDSHEzFOm3Nq10iNpZHsuxldbADsM7oXN
/G3UndGinpMEVv0UE/
hafPdm1yla2mYHPxY3nzseBtuj1DFMgH0zo9dmjikR5J01b0iwaIFRKVCZTXllEkuU7QpFWLH6Fs3qkBtXdG
/gA0JdceqJGq54zfEYluxzoYSwJ4bZXFl5XfZcsWz0hstZF1j1AVRLU6eocouQnqb/
JGktIvK0+
UboQVPWmEvHXSp0s2GEfbMTj5i8lstN1ON2lZiUizAzLmquYoIykeBwPAxmfCOeOaX7iVXo9hxR9rVK9Y4ya
+FfRPb7FvDOpivaUjJsyI4WGEI4NqjyqEcY+EMUg2EOEMzG6Ci+ZWtF+
zsMr8ovRgBQ7HA9hrLGbLpLufax0GszOwWqGPK6d0X3ydBA2yWzY4A39ixqtqHjx5hWGFUOPLa3ohcXuvF3r
+
soNeRdnBibEIFJHBEqcXPd0x0PYcy4vlR18Nftmxk51qfK6BNSwjFQ8OQDHfax15c9gNCNi6FWeurV9s7q9A
+
ls16SVO5o7HsN9X8YnBBy06q5Sm0OtzYKlbRHAOZ0fo4hKiTg3gt4nQ95UNyyefO34Oy4guk
+2
V3x4tFCPpZnHeSQCsRS3G218JhUQd4OiTRTnqbdGV3a0Sq3oshF02V9PSWcs4Xl6577ccPcsfF3rBoZKLCsW
/qrtgmXirLQrSPm+r4Ffx9lDwROV+4doJhVPTQmmBTNN5BO5o5Xk2XE/
maP1vXoWTrVEGPG4rZS6sV1eJOJdgMYbsKtZ83m+duGY2MY6sBsVIWuwRQ6zRpHOvc/bq+9
vUARDnSYw3U8HKIvOzZRvUcy29EZWt7Y/2iKGKxa6cBN547nuA/
ybnYLViWE91RTAoT1m3RaHRLs1K7sU0N0ujRmRpoL4lemxpXA8QSNbPZtwtqUd6PLWCGzkfb2irq1O4O90HJ
/uRT88Gmc0KYUyCdfddJudXJqM4wlw+
jSPezD3Q8wrUuP6lroPuz9FcFyCLTeuQ5sOF1X2iAkVyfHm6BxaM3cYdJsV5+
hsNVSEd99MZKfMyQFg6lmKVvxCUYrag8PCBxNLPnJjn40KF4ZzPAmjJSQl3NG63M9epWFRbtRS8e
/c33ej9Cr0fVRX4sVgU1Wv7mvtWiX2PyLF35bliidUKQG4vxor6+
mZzVy62bo3w5qKyhQCZgp9Bsx++q5KBgY+
lqOTbMViEGxc2qvjecakAxAjC29MCzNoKQL1eKG7cd2fmMEqNmrPgNmicaG8SiBWytxXc7pcjoEb9ejINunM
/NxSncKhh+e3Tunbz5fFp4SlbYa2xxcWarjBfg7KQrtaHffqQQJlcyUvt9kYFKSxBK/
hdE4ikeMYhdVZ1CSpsZFt/UBD8kTI7E3cMDCVqS5TMHAtKIMvmIQHM6iOlWIwape6EtRf4r
+FlOZI6OUHI5XkERzkZ+lgHAOwB9o1WJf4e/
pJQBoiWxzVKbHOULtyTGAkeTCYayvlrn0V6PLu9upWoTYVD6g2ElfBRQptlaj74mRYBtlQ4Re1sbc077CcY1
/mmxH36HMPeyBo7TejI/
DrAtlQn21vHhLVHnjjbwWnevsa72VDV0mxZGo2vFEVgfu3ao6G4ulSXc43tY0yFvm7iqEEJDscLMEjonLds8
+Ljv9ll6FAnn8x+
XoXLBySe5vl8JvTeCnfwnaWmhXavMfN1V0W9I94WHI4nwZpQZCbhYVJFmWA8rfqcaoayYH6sdI7Aq4xhvBSX
/7ylIsNEVtmg9ErRq2ieaJfILpfWU/6
KRkSGTzM3ojtczYKlBmx37GTktATPrwESEiSkYQaN4S+CmX4TY30UFBPW9gQt4lK7v+
U0jyuBOPCWuu01w+4
AIIWzFu5ZMoAxY9dmbUGN5G4AO6txGDxyOl20nMIGcy8dfezaMYjTQ6S0qfn2JW8AwQBOonaABVw95DMoHC0
/UegOEKZlnHf3jpfkxTcXDIZleTW47kuOFzALJ4e8wewXa0Nd/
QatyJWVIa57M5JHWiiD8DNAkPT6N3UKbJdmpsbt0rtNfbXQAWSbnJGd50pNskgs1niiYAWnZ6TezJ7D2DoTS
+GItwOM37ABxlbqUz8awmjWLSrewY4dmqYRokPHAppGHA1xP1AMK5QTr+
TE930bVfCW8roKnbC0UQFbt0PjM0nc3AlTXsFZLeifKdLt19orC46giAAQopjpUS2WJN0gPqYODnm1hUptmI
/ePf/u2halw/O3tnxPGbt937lg4VB1DyPUOoYBqeloxTetjVdRTt5u6l8R+
szen9h3KNuFY2dQvAkD3p5+/MF2WwznnYkADMnc/vDDRrlIbr+yjCIik1w+
fl64zNrXPDELxmGK7oPeAMGDLpaVu9hpyxES3A4nsVsADReJJXXcRhESqlJeZ9JJps1W0nTJM0j5o2M8WF91
+vsmAXex68JMlrQmc5U0B/0JKb6PQr7vLqTXZ4U7UlYfPzJm8N4R3cxtFcqZfw2gl/QtL65
/xlZmKiHmbqSFsfzMM35XOWtqtudA3SCi8aGjbsBTN6m+2aEze9HJVE63XSby/
cFzpq6QUvcXpumN9B725eLcr/sAfD7v763yD9V+
IvfVgvjAKtjcN1QeO3TYVH7ebZCpRHMjqfpmDxP4y4wnOTvmFk/

```
    kbETuuNPnNHPaaviTodkG6kr8RnH0khX517X5tQep+
    YwyuwyDxYSL0fTmygu3qd5wzfGNHc2DZT478NBLstyMe0fD9seJNY6H7SNq0eoZhc4VtYh0WkRWc7aZk7ojj
    /bCwaGmQLao6efCoRdYZB5v31C6SDdfN1JJZU12Ax8wVrECfwV89WjVlaz+Uj+
    ubKSE3wjl5QwgGSoDLw4vV5gETuiOF2DQJ1odGhkzvUVbaSzxxjem9BIoDfzxUW72THrzlKiQ
    +alHtWwK9++Fcc58/7Hpf02ryk2aewAJY1VF3P+UuhnrF8wJ3fEsZu/
    SeKTJfBwrg3gBjE7pSLy0JrtPzBMpDvuAzGAf/2wuWihzQ+L/9huew2pXs2rPhufR5+
    PAUwbHbb9zDFV0WC9j47yxCw7Hizmw9p4CW1VJSqZ0QlRKInRBk9XMg/
    zhNY38ek038y78p5v8c0imSonqjsfFyr8kHhVKx5t3Bt3Y25Z7AGkOd8DheEXpY6MvCiItZZ7jN7JLJ4Q4cR
    /cLgrmb8l/W+afSufLTCnxwSUzQFx36cXi8hAzYqAfV2TTyUyo3eVBQLdLsVAq/49/Ko5/
    FDRowyMwBCuudKZfV0TG6nRy3/fExzv6ALw60sXTxvJ82MWzssX/8zK/
    vIHRQRkc910Ahm76dGkeR7Gart9Glwn/nYbcMyVG//
    mPxfFPgblxSJMeTp0tgSXs9HAyryg4lpb7HrFttQe1ZbpK/w8q/4SBfMvLxak/P26OH7SH+
    VoC4p88ml0Yb/uzTXtfnTUY52ryP/lkHC/A+BV4BtnWug/
    doB0JtGJwnPRjZbBJADhnnEMvI1v02mqZ/43
    Mv94I49L35QeJT4c1F93ch3OY81b3NuPPqdBLlYPjZ8M4x3+/0
    DcZlc5KbrYQF7rFG0OjV6F0wr6ED8pAfXFwfUbIcYN335YZdsfhwppJGzVV7qpafjScAbDOcVcUxfthyxh
    /4pZbhlpVeb8BF0YFynzSD6ilgsQsV6EMDuWpsZ15d3uC1f9nvn4Z6m2/g3124El
yN3gWitJ0j8kfDOPg58PgJifxnj5xO+iRn+a5taJlJL++
    TBrJZ1kLchrAlW6mTEQgPHH3rWsEqsP76heRikTcUM1dA0fQGQfxtu45+WPh4G8iQdZ4Y/
    monOxOZOPgdxtwoUSWd1dGU2CxBNJKvRbRPK7djYrcNWdu9ov39mHVkHSVFSiKOLjP74fCOHRChZB88nrqm5
    /ndKFyp2LQJNNhcbu76PKrc5xGfcfXw/Ew7yKJSBt39UUHYDpAUg6/MKr9uRJf4aI+8
    TVU88bTmdm8hBoZQ6cqtTDRK5z+9
    nwuEtVlUU8ERAbvZtpZQ5pXdSYgBnJfNUE1zMuJpWO2t1AEE7WVe95vgFQ0vk18OzHLZCIdZ0Szqlv9
    +vHabKgji11a50d6kNzlYjMOE7pbF3n+FPhB2sypoC//2
    q9CadPuQDU7qXMt1JxiSYhF5Husqr34TD0QndAeC/KytB+sQuvV/
    rWV7pxh4DB0SOlYl3aipvmazB8SD2g/S+E/
    rPIy2UnckTcXc9kVX0QMPscypW3z57AqWiwbLjbtFnKhBcxsyPxx/WHGd5A+
    CvLxXDrmWNH6KKtd/jlxPDLGbYAT91On8COdMbQKXg+GFMic5fPkg3e6EJpaJGtW/5
    BC9qGXUc6TbzvCCe9Pvv0cQLjiks2f+oUuVvhBud3YeuIP1nwaCjamg+
    V8PGSR7sh5rFltdnaE+A11m5pS2Af7vt07+
    Xig7yDXD8LPQJvTmfjyN1IUye1Ba7D7x78dVcQjJ7J5KOvWCWMc4BOHez+Z+4SZ+7D/
    NnofsEjxkArOOLQnPgT3dhC7Q5XfVH27V14s/s993/
    rLb8H4HBTlFCF437STBoZeqetgTO0fB370Oaz98MHEpxt0q4iLl9hJS20Ot4t7i8E/
    JfBh5wIN7CPTp/EgzWgqzbL8UijEOZBRnn8DRSazISRL++
    tOdjINz6CC3A8ZdCjGK9sDnwwfGDYJca8RvBl50DB+iN4BVohquYXwwl4jXv+
    rffzUfXeabtppe/GA6dzIuiIAjDbL4fgeOn0VeistcZ5/
    AajKWxQsTs2ihsno3zt95Iwnp7CoPPXx4kTud/NRxGg8Fy3Wh12cd3boP+B3t3op0oEgVg+
    FYVm2xuKIgiuCCI4n3/t5ukXYpFMiYSo0l9Mz1nZpJOd5+T/1
    RRBdQfU1qLY1b5JOK7n4hTsQELtss+IiYSvklB+HaU72WIh9F/P/
    JR6L4BbaIw0bEijNTTOSr8+D9HDC+
    PQCil5OnPoBJaQWqp8yP1GU6AtFy6FWCZ72ZMKr8HVjXgRZAi0YzwpAgosDRqN8bx5DLSeummj0USoq4yLJn
    /V/Qv/DgCAGmuziwgjTe2DKBW+n3fuBTothS6jYh6P1ERmYT/
    ePAkyLuGsKmsKIZpTsfj6Wr4ZjV8+7fp2DQNRZFpwzxZRC/8BKJECUqorSsX6cVZdA/
    gPC4Vv+kp/
    eoh4gRgI1UX4uKZdrpCZ3j4sRR439Vxmyrm2LV2o7Q3yBf7ebTNZrHvO0nSD8Mw6P4ThG+
    SJPH9OM623ny/yAe9dLSzhmNToZXkn/94ZuF3IJBulrPzM6MECnpYIOOVKKGm6xpwyf1rW/
    UJNmDoU/gpx8DhjBpjd5f28o2XxX4/sDVVYvhJTFJ10+j78dbb5710544Nwn+
    BN6J34VtRWEoanoTTDx5Hd7yeNRyOp2Nr2Vt4md+39
    WS2H3gbC97RLy0N5AHi1WoCAx6rXriyskbvf9JDEtiqVEsXGXv/gQxPf1f+
    cTnKmGGVpNpBcsi8RW9krRTRu/AA8kjDi5hSaH7nKt073SAIqotmur/
    oLclXdtcpwHiv11JniP4UHqp0OTK1RoP59pB09Q4rp40M78EQy9mzjt5NDtv5ILWmFP4hInehfQSGOjK8mIN
    /BCrjGfF/8dUZ/KVQZ1QACvq8IyOf0l7Cg9TSNxw09ybOUExcF53yxiyYvCBM/
    Py9Hw1RKm4eUVolRJigbovFAC7DtYwvE7yTQDyhaEUYLc/2FjQHTV+pfYbPyVuTfaZExYLZ
    /gY5d5DJ9tPrFPuonahLRS2WDI3CFxEyPBm3Q396u6eMlzm24P/Lp554+/vnDdOh+
    lm5gSa9ODAm3PH9+E926RDKmoX2sL3y/lGGi2e2MPwdsH0jrtvqWK8UxT4XnyuLls9z/83
    ij+h99Hd93rHQxGJqF24E4WIlVfcDMIL7H0uggS++P34oO1kPpC7vcjpahI+
    OUnrOlHPFUO7cC8CioYlB6BwMcNPpe7Eq6d9gvwcuTma+y/QeLn2+
    cgUsQv3oJVhm4WbwuTd6CPDz5gsn/
    Ggw3Pk7iALnnSu3oSvy88GQxG70Np6nOa5QPh479c2fz/
```

cctI68ZM9XHqO3Fr4tvpCA3mdpNr+whKxC1/VL4UcW0CLJ6OE/BnxEsau5/5MT5eSU+
QbR1dfcCCvY6ruHG9GFK0Lt+MnIHHqxAUKnDHM+4godTTdDvtOPNt6Wz/
QWNNm1LMM6eSYwnDhaK84W2/
GOpqTD8U7YYRPl25JWBBC9SAzZTqdmqZpGMobWaayYpjm2Jr0r6b+DFfpx6Hc6MW/
LHIeux73FDGwC58rfYAXzHYG+RAI3EBe2ljFcAg/iJ/
atPa6vzNyHnvXWwMAEa0Lt5Ye4wVjkr0FCjeRJ1ptUF/CTyLHodxR2a+u/
PL8q98zAEC0LtyCkC4esQBtVOcpULiNEWPFDH4M+XdVPu/+
hch57N35EMRZk8JNzM55Pojsk6cibypVdQk8Gq98nWl/KPIzpm13onXh/
xHoFbfSdUM2PnHCsopFEoGfQUeHDv5ZnXj0JPsdwhMjoOFJR8ujzsgCcvNPHQc/
v5Mup85L3w7TBslJn2HLQ3hqOzxj+
kTLgMKtCNAZFqzh0eSJ8wcn7NcwJxXnJP51hL5prnWLF2EPZwRuR8FBbgCPROXUF5UXMH+
kiIv1P4soo+gQaMFiKMtwDaEanrFoLjXP3Qmphx79zLI7lXfxn5+x10nxWhYX7H+
SsVEvFQer6wl7fCvdTtUY5IbOBzmthT4o5NaFx6CKu+0giuG8iiGiGrmyGNf/
GqXXKX0bzK6mruh44US213BetqmjD6TcfumE1BAeQRlvdBQ+oG/
GYm3uL5FHOmLl1cXRmEDNSsIjScJNqjv9MRCoMvqIbCFXQp8Wokvg28nmpI9iMP/fcb0/
MUXrf4W8rRfBUNpPoSbFI22G4WrLpBwoVMghvpsDLYXuanjhwPcixjqTROW3tS5l1jM9OCx8EwLKoeFboJNC
+v4YWDCuHHnewYsDfCdlldtiMP/
MsN7NV2LL7ZcjYCTNRwgMoEq28SzYMklPq6EPz4X1QSmGPn/Qqru5jJmo/
LOts3hpgvB7EZgG+IFIgQqXNyRlWgwUSpQET/R0AgROZIiRm0O7+
Avg3YUuBvMvDusLVwF42nd3Cl/EX83+AYZ+
tXSyQXb6GAZ7e1S5FN8j51ByGdAVG7k9tI78G8xnYjC/Z1ifLaci9d+IL5E1YehdWWy7SEK
/8gUj5KQe0HPnfSyYQLsIASDDQSAG83uH9TAfUpH677NT8X+wepVuccDPvHLoCyzoy+8
REgpKggVa+4O5svNUUXk799HsxAz+VyEAIwn/n+
pCmTzjRTmuP5BNAhdLLAonBACq6309OnLm08lBzNnba/0
wETP434PAtQOVOqGflatkmChQNu3wqLa9zJ3yTyCyhEVssxpNeiEWaXK7mbt5KObsLc/
gFy4AiJvhfwECsMCaf48wyhsbCxguoYzuC1ll4QJo4cseKjt0eq3AOYGWUACy88Q6+
zekrntrIt409/
oIyHOssnMZgFKAaaRigS9DmdLnZXUkNoGCHdaw8n9Y0NpLHpV0Ji7NuVZb72SpLFJ/
dRTqnfeHAPRUkOXgBcMdVKy0c10dhmG4X/GHI6iOH4tay9zo+
eLSnGs9deb3FJH6S6Ow62CFbwLlFckH5BIFKkaXvFSMBmxgyXC2/
jg9W2kp82meiDn7d2KImOSmSP2VEb+ayEwBChe89IYjFxZ4JGkSm/i6Oealex/
Ux3DZTubjfYhC++
rrcvuxSP1VURhUW4xkIADl0psvq8nloxIidvwgdlZwJnexCcOQtpH50Oui8CDd+Uqk/
pIIjKud7AkQKKGwVi95ulCTI+P9xgdpPbwkbGGjZNVG5pGNwgPZ0VCk/oJqT7JIi2ufNOx+
FPqweypdVZHhUlpMDDgbMGRYw5DNlRYy34r3xjycvhWpv6DqmrsK5EroZngpdA111mnlXZJwECaLYEThYqQj
/dnvspE5jcRqQtAYIYlSyBXRnT7w9BhIuGJHeM6zihwbh9ZJXOMVwDkvttjxpmGwg/
Rs7G4W+6
lEDADLFF3V7Ia8Wv0FVyz4T8f7WmYQoE5Q2TsFDtjiFJO4R6UgLkVmf8ofWsCEam/
DgIZlqnGlRc048X06heBDNXz1TdGuT2GAnkjFZ+Jmq3vy5yCEolJ+4/
TPQWoSP1VEFBCLEuAVkP3+
LTbbLiN1sFQxZPcTZTSh4fWZH6wJUQ1Wo6VOzMnc5H5U9D3VFyqvwwCloZlHshN1/
EM102XAHYS4ruQOR3Xi2uFKuZq6I7p3WtwuS5ugXsSTM/
FqtzLIDCQsIR5QMs3yao89FHTV5mogXo8ktfHzlSbUyDQFp552hWZPxHWTcWq3KuoX6azDGTgSrfAWo133pz
/MmwcC1W5V4DARpiRQy0shTHV92bTENETUVkKM01eaoqOyCkzZfBTX1xRuITkvyxmL+/
BAJDDSt8oHwTvRj69JZbb7I8VNTtYN3m71GOROZPqhPJYv7+
Cij0ahEllPATWTj2QeipioGGb+IkwmARbdRJCqSdzCFXUXha6kBcqr8CChlWhQoh/5
bqkGOYE2gkmwkeLVQbHa2L0hjasRPPrjw5ewdEzN//Y+9OlNNGgjAA/9OjCwkhBOK+
wYA5TL//2+36CJIQYiMEWVvpr+
JyKrFTKZmfGY1mur89hZAvBV62nFyjjlvMvf7qsPr5WXcfs3svYvHtRSMpF/
sDWAFfcrug6UUPB8INKv2vaI58lGf2Zan9R9BTabv87SnU3WzS/
csSMY35q6eQi7DhyOYvNe6hvFe5Of8xnA7EN6ewcvi3TAYm8hAW+
rxtauXWFcpRq5DFD7JdQXxvCr7Nv8c++mZu0sf8adtqrhRKUdaUxc+ip9Jf/ZsjLA3+
Xc4L5e2Qa/C72tFHSdSRs6g/jmZ3KQ/avrfM4/R8mnndM/
P22Wm2w6DsDljVHbL4kU5diO8svsP+LdGLwhWEl9Prm6dQivmmpVL7z6TZaMn8/Vsj9AuV+
t6YuIJQugMn+RMWP5bmmS/P1P9f9ZFl1kcw85K+4yLCFytZUfZrjU6RUijD60vjlR9NM0/
rEP8T69V/azjRLnA680U7ZzReF/t5nnwLUPj4GPIGD0AdV2L+02luLGX/zP+
i3Rxygjv2PWQRIi4W9eHA+8z50WBjiXIU0N3LcF4Bmnm3guyK/
bMU8Fpj1lrXmFnXNLtrNk5LZCkKizbliryVBcw/7s3qJf+
bNHAk5tWg2RnIotwfpeAtPksta/c956dxxK7TWPcHi66nPBNJygo4V802rrRdcYJhZ/f5+2

m5nLfXMpxXhmYet2VQ/3
MU2sNEr2vDcKaj0OHGYBMawxfT7yoTSXWXrzJm89dl83ZJdb0sM5w3ZTivGLtpSdL/
EEJqi5nx/jGNmGuGMZ24+6mP5gFJqW3v2b5s/snI5vv8OfJkOBcyqP95Cr5xNT+GPd/wv07
+sIU03+
YMYwDQO6DpcK7GCPdQUE1ZbK8kpymVpv4EQsg5I7C92LJmZzOfraCQFG97zzZrUoRDftIn9fuG89VYhvNK0s
+XsVnY/zJ61o24ZSSBjobNKX+
PKRdP24oCugKcWiKswdSE25J1OgID2Q62xCw91p6MMkJLzxBZ1OusvXaN6bKEgp1Heys73a5iOpCf1UhHkcw
/dZTSDreSrqCf+Jrnc9rvTuG886MRcVNljKoPxGhpfksPL61WptdNLEzIZ0tosCDosTNVJ
/1raTTm5vNud1BQQSzX5PhvPqMqSVJfxaFts1fnP1rHR+81Uu/
kY564DQ249AHTpTohXw76eiNL6cJ7kvxabu/ZfFXiHpSEvrRsmdUhj4ARf/Cu/
bGSaa0Nowm0drf+zy04uYo6ytJP6SeiDWSj9A57KEYAt6kCfJfw5E+D89BeDUSbVCVOo+
jCugdk1E3QocPmAaao7gbkxVdSboPldjj0nfiH+PCQzEEbyyT9r+
IHnuS9Geg8BzQHlLUx6huc8Li6GHL7tbDWT3kTBAdHwqx3mA/CYJgEux6gEIBirCU+
hJpmvXHr8o+hAheZPoee/wjdM1LXFC42HS6dQfrnd5a3S7OusHtpCsAZvfdigrmnICp1Gw
/01pf+bPqxb22ken740V8tlFXj5EskmOHs6nNog68RGJ7bjbpdnJMV4nQF2GiLtUff/
kMtHZn0Xq/m8/nu/EwDGz+ULmwD2X6/
jjZjsdDQpYCDskkbwfBNpiCEHtxcpJejom07IVLbGpwx61Db1X3LNMkItPy6qO2f2jtwhpz1fYGuzJ9f2LQu
+bZTJ9xA3qq5rVbW9VA2zzyOQn3CMV3P+zIsbf5lwqyZjTFMC78sNGdEINzLxItM23+
ZvVj4b1b76FQq6o22nGd7FAVrxgl9hTyj8GITfK1OK5xNOWt+
b9IV4U2m7R80G00Lv4fqg6BKUTeaEI9CISdoC/
mGnNZYEWI7ztBNKNyBgH11Xq6laN52UYC1rFLU9U40qT+IQsRJU+
RTM04x0qUoxpzhgFAcYSSdkD9p7lsoxhrY1Uk6zzxJ+kMQjunwmurGNL/
BKbPUTZQa8qUAdFfLVmmR+
klzi1CYN61O0Nn2JemPoOC7nLTBjaSPbE7SRxBiZsgXTiAUpbAwWLzT3MRd2hWaEemmJL28bNuVWvfmu0KNk
+pnC3UYWu40yauZSn0Hb5xuT9dnt0p51Oes/
hhCEIBfly9PxsQyihSndAThvi0XN3NswCXz1JNdkgHGw+
CxShoEF1XpylbRTKIPjV2XKk0xAlUeZ+bnYroGTZ/KkWXUv6a42/OJZCQRsWv/
QJ5RCsE1fGBqIkhZDTap1bS++vmj8Z/G4Puvq2UVspFGPtK/
QAuKw5oSylcKzOBV3LxveSCMsap9k3b9NPnKDDAxRiJo7GR8694stHkvOzvfmgsiLV2Uk88yBKIYw5TW
+Qj0yH8xfkQGiFtuEUHs/9huT8bP2g/
qImDtVZ3ZQluZLUlSbIC3Vj9n64PDlMSCCAXusoqCOF22PRwzZ+EtoBV0XtAFFG/
NAsNicoXKewy668JygiQKGQpqQ8NhmB8CCEUXWqa+qmHHIphTDjS0MrP+
nkXvRMVEhTKISOMpzH3DYID0OoV6ib1ZQk6SUoHGp8aTZCHuXr9LFhQhnWTnIeMw4g4KFJr87sncemJL1ku5
+bat0PJeUIfCg9GGFToCk+6kvT7Ed44w+
hbgMqJp847pSaP1UqIl0YeyUR1jgRqdnsQdyNMOCt8zXvdpXotL+4
PeqdK9VDKa6xAeDwyqzN512wfZPH9FnX7L32Dr9gpEK5KrJRvLXyRUyylGB0QnoDQ4srQXHuVpP8XpYq8FjT
/
gUi9PmacR3XY9w7pzQY3RF0JafVLm2f9uxIVepaa9bShzF3zj6ctdYrao8AuvGMLcsI5iuQQtaCNX
+
Y3TGiK6DPIsnugfAkZFXpJkkz9yXpFwigaX8z50nIG4SzVv6oboZ8lfNy9Vv8r6AH0fSe4nA7Fkl6AcKzECp
+a9bNTNX+fHXXsIhUII5phFSgSFp1FYcMWsTUl6zMQgCuJGSh+
falEc2xSF9vZ60tlpAkSEhLrBX+zlX1395DHsNghP1KvcwufQkrX3OOddh7PcjQdSV0+
srllfX/6Y9wAoUufV+9a5C9ugcM69iEWKnj4354TKBZ0jSfo/7
N3pVuJMEAbgt6qzkSD7jiK7gmLd/919n8sQggkmohi6+/
kx58zozJwDvvSS7qq4WmNdUjW7QGrS3alkCJ67QwD8CtjXeFdyV3DTnW1R58/
uQPjEPmA7rWkPvr8hRHXJ4GwipKKZZFC+
M31s4xX1hqF8ULVibXMYPd02hs7nzMH4TQQdF0u2N9sbwq1kewZnVoFQkqESjlq79axar8i7sKPWHorlXJ9S
+z08eQHccpcJ49rSpcxX5uVjYnH/ij0H4ZUvRUd2O6Yxn0WkLOvtKmao5E16C8ue8r1EZhJ
+ipmD8MnL12417VRvDbPzljZFK9hWKRd7ay0HNmdy4yMvVqVjhzwmY8NsI+
nRuSQjNTjqn3hhJxnezDzoBYBD24sdsJwUbqferNuclPhMXB13XI0phFeZi3ATZGY+
vlxIAYgLR0U1Wd61yRN2vOU8gI4sP/6AQjAtoiKaMTro7SllPN0er7SD+
Qm03BsCAO4zgrr1JIuv9PDWenMk4sjk/
j2pdJuhV0VVobGNlxloSlEwWkee5rotePf7DNdjF8r7iVIYNP1SN8WHS3d3Xg/
qKmO1ztasY0Am+6Krmmbn1zsf1m5U0GB+8puz5c2AXvH2Hev3Fb89AiT2508KqfX5+
LQO6nkdmjE46fTpjquaILUXFSV8nv1OpIQh73HKyB3XlV9og5EMY2/
NwqQIwLkKnelKf1F0Tk46JJFT6KRUjMjhLMGK9lWSpzTzkRfZ8ewa1BuEyFqIxA5NO6KrkvH2JBHcqJ1T6Lg
+CfnZ++
eZNiBcit6Xgx9MOwxLU1FH7U6Tvuy7p3YgxAjeuiKHwhsQCnBvbfn2dKqNi9HzXkusY1jSh3JIyROORauC4w
+mEx+NcKQnmLg5wno7kHcYRYgIzEYpp2ZxnCkG4oEj3d0LNzEl6+kS50+bTe+

```
    Cq0mkGFUfJ3kMVhPMNbc5LM9ekiWhODc1J+k5Uam+LMf4Z38ux8DkC4N0OQtm7ww/
    o2pifMMRlebqv0sWZG5J0QjvMaELZiPCu7afc/yFmpsSevdqcX0/3RutDGmcb48JutP/
    crdwYknRgIBk1J+6eXLzqHn9D5V/3ACI3lNjq3KS3tR9BzuKAcGEN7ZMeLMxIOqFdy+xNN
    +0B4E+fBF3EFodfvMVZxnW7QD+lDsKlNXRqt5iqVjWjWRNjUZcsQSPCOJSkGQ7wSGJ+
    C4Rv8zo25+U7y/
    WkafG4WNOQtmyM6FZlDuqjfnIfXMmWcWjuSGx0xtY724Ov5dp0f0PwNvqWoHg3IjNm7wwMQ8miapJwd1waexj
    /YYI/wMDNg+bvzAZmJJ0Y41X2oJ78bReEp
O3BV1/wXTub86+
    s8ReIgbXmu6RrGIKBVignxEn2cMSdSCxuoVgI2YMyOczwNxjoP4jOb5BjzMEZYrRf8iQ9pV5zO
    /4
    R8GcevoGw1H5z96rHHQYPNH6HlPhLMzbkADBoF8hpSjZ88qJrrQ9CYYSF5lPDK16jv2Ogq/
    H2u5JwYVDScTP6xjkinoh/5hrdsxVfy1Ze5hgRei/6Tt+V3EfGJB0Md+
    ZLNiWD9Mff8V96BqEohq0Ql8sIhL/
    D4JnOD9peXFPW6QADy45ku0v71Kuqo06MBTGmYuVxB8LlJad8+
    g7qMjDkIdsbRjTLLs4e7giMI64je8ESVPh/3ImVSwWEv0QEt1XXN+
    otGISBpwfJNAUYCXx4Gtrvgor+f12Nt3N/WET4Wwz01oGuUXeeDBrSAUL1RG/
    UhzaIMiuLPXgohuyNtQK6+
    HMEtJ9D0THrypibbB8IeLqXLMEOIOxRSw5sURC7dsM9v2kJfg4JQHvXFBGlW9aV3EcleIUvajzJ
    /MxWyZsrXVEV+WcCQhGuzq38DlzxPdX0qI+7m0DHrN8a84jtHYHm2ZfD6wf3+
    qJNIPug31VBxXI+ECs/VZIuoATAbT9uQ/mf0irtOxhnPMgc1O/a+
    ECNivPtkvhsQAGTn9UoxZC+
    z3pvuR4F8kb9T4d1u5rDODwPs946f4Z33JQDHRTB6NuNuGIeSnd6y+v1h4NRzZE9dd1DvL+
    AecabzEH9EUwARUpiQbGSmmRPvhbl/PkDtjTsVXvt5XA2XTVrvohc9xh/58E8/
    OhnvFdBHwCjIwdq7SJBZ7sRV/KeTAURe1513Gs/NZ63nXqc+
    OvKu5Jt2WZNF9FbZbxLzqjPaMq3Z+4utG8P8AvC0s3d07DrRdVxe94arO4DdXWBb5X3w/
    QX8TDrSKxyNxIreEeV8WhruBenruvw1mvio/
    GyMbltxoEvfd7VDYzUy6rOmkyqX6ReHKGn8fXmY1pvx+XB7Hre/4
    EfrO58eVXuo3VBBCNxS8nXwl6RoFNHrG9werhizK7r9bq7aScscc5FOtf4afoT2vd5OlOyvZr6
    +7
    ZXOaQfIWa3152MSvt4dX1NC6SfxGslp4UFLgSwAQ1AfkulJKfjfgKRN580Szm2L2EmQr8uJ3XABf4xu0D
    /LjXQYUhPuFmXL+wVQ5fpIPBAne5gRciL7QL9+3xXnyF9j7urks3xmhq+
    yvkwlqFkCQp0nyV7leUcaqLdkA4iAMNyjevPMBXD3SpJVRkWmbgPS/
    bhfWUCHYd0EAHjrVOirJu6TAeI0Q0khVoXyXm7LtYZ1FS/If0VMeBNy3OMyjd1mQ6A4b0o+
    WQLF7nxS5nPSlwDf3xVx+MKYIK3Kc18r6Pry5wDMRqBHLmHi/zWYp1ppeeQ/
    hH1XqcsA8HO4KTDRfXojfAXRTbi5uWZm12typO+
    SS9T9yfVNznpTEg0aVO7Igv0atNO3M93r/MPIDFoWo75e6jv52keLhYHrfFv4SK/
    qc35D1BrjYd0gAnLcuzY3mr9Ov/H3r12pRHEYAB+kwEXkYtcFCheKhURRfP//11
    ra09tRdity2Znkuf4lS9zNmZ2J8nsxcCmJb+EOSO/mbgytBK//
    JPB9Ujq92mv8z7E6A7kxfFFoZO1trhSjJE2JpzV4WHJOim/
    JeXAwDQT6RUaE8dL37j75j0nxnkdKqXjHABQImI8DpbFdpBDj/
    PSZN3Un0AGrmrwvNykvs57MZjAyG+S8vXalbtM/uZ+Jsz0O9aPR6mv814MEPJrjj2hl2mZ/
    n3ejO61/Bc/y9Rz43H+
    yvurCpRn6Q8FT7Bd8JAmtTguSUljln6uYTTVp44d33qk59f0kriShbRLYV9xDbojLjn5f6jlufM4L1uQ1iT9S
    +c/9P8Ywq8c17Xk39j6dpmsIAxplypF97Rvcv7qoaCwMpHaze3Zze7N2DGHmYH0aQZxO3/
    BIedSM9M/Dd8/
    NoIO5QziykdDBuG6Jp5Zt3r3FXFGRtI9MQFrqRnnwX0afR3EtlDqhtZN4RYRFEUSv5LqJPq0O3YbKCGKiPe3U
    +EtakAdLR7wnl6Jn1YHbhrARd2j9cxhynomeG9+8
    b0UdP0KvgKGXdOVGyFbX1FLnxViJO7QgTzBF7zg9yIO/
    pL9HOPPS1yp8sXWTN2kOMbn3lP4OQ39YrwnZCKYcLUXNwFDVQk6MqbgKBFnAlnlPCvD3pIMieDNLNYJMrb05c
    /5sqQLY3gqah7h/mD0xH3IXxw/hdBZiZaVrUOO3Rh3frRWnYG1Z4/0
    ruwMMoN7RTjyo7UqTWAMYSZa2qZqjndi706tUpAre1+IaCo6gtzB/d5Y+biJSmVz/PbYMXH
    +Q+gMREnDWIWSH63VxZ+mdBrKtY1IZ9wWySZe8l4+
    xr0frVWsdwKA0NxIkP7cxIAExo3oCPbOM7di+FVrFXtN6Z21hF8d6oT/Etek0+
    azFOb3OZSG8eRHa5XLHqnZXUmQF+0RGMURgIi2/
    YxJS1QEOYN5hGZfXOWy9cOxhM9MSCDQZBFTdxZjKEp6Ju6824lxJU5b9ghCUZN1OwziKge5kGJ8JGdZCHMfCI
    /kh5hOiRnf2qIjs1aM+C/
    GWlwNNGYAITfCaZAgL38bRIMxDJKbV82UhzDx4td6CMMCkc44a0iQF1FFOmEpOhrmeoP/
    wngQVxN3AOevPolyrLHa5j3IFIYxbjNxdXFFYOTAGLXkjXAaUaQrHeaGYLm3hbEUVx/
    rJhh7Meb9dxe6xYIwFh0XMIux8G6WWlkegbEH4+
```

hdrLQ68aR0tdb0b7Gs0U9pfBpxHxifgLETgR/
knUE09fKMqyAqvsAo9kFx9X09v8XlTraYRhPpoEvRMbKa0skTev30R+DdB+
iyTYjmNV0vv4yj+
Wb5yttTU9bugnd9Vml88D00JaUznkVH12Sggz2h11K2A036lBVkq3Usm3dCpyUqBhZT0uPe21Pr6cNyWMZ57
+NfRfMKytgEyc1T+
ic1PaHXVXjaGumEk4F87DqWlA5GX3Lyl063JKdsCtCWqdzPsk0I5pJgte9xYRLJCpWIL8Qjvb7GW0pn
+Kvs1IsopY9FxSqaJSoJYeFhXmvtLojwFm1ktzCLJaUTzpVSurk5kfzgCb3ewgYgvPEk+/
SjyVeMr0E0jKNZou/s3QlyGjEQBdCv0cCAwUsMGMcE7GDHGyR9/9ulSDlVzlJhwI7Eb/13
BNVourV0610ErHSGfvB0R69n+
mfbqp6whHSETmM70Ws0ewk3CuiHLz51EPBDuPS1BA256lWvaIboHQQcqw6dQn00HwZja6N/
SxPSKxxZDnVZdekLBXQSvfW0e7vuWyuRp7Yl5Lqw9YlmiNQpTvhP2FBhbjn0ByxJzzv4rIDuU39Fk7sDd9Ey
+/8
I30FNC9mvNMdIRbyyGuicboTYZzE6d6A57cPVcrs6NSJvpEAd2t5phnoiPMomVQn5cx07tjE6
/6j1Qf8b01Ver65g20FdD9qj8yfcRhEC2D/
oRpkPbVUTmLY2QRHTe2E5W2tDfSNHesuSNaowPIE9KbKdco7SPcK6A7xrTrvhEWls0z/5A+
VDmLZ3T7TIPaMjgi+x/u4aMCumNEVS0vwtpa06WZ9rq6L0NZM6KLV01DetGZj87WhLL18+
uQHi25+oxvpNTMXV7Ee8LPd1jbDtRpppULxXPP+
ozHRnn0gXpMd4V3FlSI7lqPMkzNavuvyN+
q20dXnWVcI1yiA6iyNDY79XzCtlRAd420H1qW1xziV87BaqN6MHGM7xQ9Z0gf+43
oFyaeUW66b9zWll6zJB2tA/1
xyha674Xq0qK1pe24bTp9E89YU3dgGi2905nT5H1i4hvpZhxQPeUI6V6347QV5x3n8drGcbT0ruDSzMQ5zgs
+ZQhpH8Y0v4YD+
yXKS1oN25jZelFqp65rQR8M3GP6DXV3w2uLL1neBM6uitTgIY2Fw14zJC7nzDW9f7bmYl/
X083/Gp2benxbmr8XYB6xRWB6NnkP5wnD+nRvnXgStAhehl4mx4GjI5sB2U+
eLFFwMKkBD3aRTqAywwh/Z54vP5U4cikBMSL9IBlbcmNB/
Bkqj33QhCfpKN7ZcnVS0chPeDJpAynvBM94NxSi7aGHxX04FopiHeXAkYnltwHP1XpAStl7qWIC96QDtxYtN
/IJ4wF4LmChzLwhx7h6wU06+
t0qFa0Whzt2XjW2l3F2Zu5g1K96Zju7YkptfwI0Ax9qkIMSFLRXWlp6P0zPK3Eszp53nCFj1bCvl7srcxazP
+AVMlLkX5gvvRM+Tu6/AT7dlynPC2/U5T+5+D36qUC0Q620LG50HS+56
CHYBK2XuxXngnecVLi29Y7Cr8GxSGuL2ExUmGe7MfAW7CicmxflEvEjvXltr6jPzImCqR9ELRFzZkiUHjSNw
/budDuNGIYC8JUNDAwUBkITyl62hCzV+79
dT9vT0nQ7Q354ei19b4APGlu2LAds1Bm0ou2bItgPNbljB8wECy+
Ls4n3CZKAuSbX4y60C5j44ZpNS7Bqojgu6g7MAubqTBrtedfuhwaK42asC6CvJPjhmlGtB9q107DR5EbMxXG
+uWcVbBtvEAVtU5g02gK06kyJ1kt7EAdsEvPxwzbB72ucWBbelJjfnPWAT7LvqjCK+
qxow1uRavEm6H65ZRvw2UxM32KIewMp7TljW4m1Xfs0NNj9gAxBG6qyKD7SBLugMNbkVa5Iu0Hj9q2G8d9IR
/2oa8dK9ocXoI+
eHUdD3FN0w4s04QDDU5Er0JD3gvafohhVPxIHexGo0KudV1YC10rtK3m5SQMBL10QqMPIU3Tbq9
/0Fd4Umt2RM0gUDT9Htisy12011L+62
wcdTd0P0YFa3qt0T9IBndWZFPfEeo193T8N4ku7PotsWiQvjAAhuPUmvQ9Bfqb0Let09qSS9ZAz0J7
+
Lbtk99V4cELDU9PheZgr4qM6sqGfuCf2aflKmy939FVXjXqhT9KbeHjnzfR3bY3V2FTvyGb2ZJH3ENmiCd6U
/Y4h6BqaXLvwCVgps4w7nKZbwY9Te4ENl4uY1nBfBm9ySR9CiqCWy+XsWxD+3
pDw0l6D1QE07/RYtkyg5W7NLHPRNYhMuBFnWEfMwh0YNDS1MhKZvx1Rdu6vK+0/
CSg10S2oNKeqx+v2TX0IMwbemlow/SJFNx4dxnLtlms3AWTqKm1uAL9fU+dWT3uu+
gXi5Ymdwsevhdn2yiDw7UvArpak80uM4Ktp+iGHT0Z0JtoZMx1v7ftnZ4Ni5NMAl2a606+4
gl0wd734gwbMnd0f+3Gd+
08Ls79xTyTCR0I6GlqRD2fAyp1ZkX25jIXAWNNjucVWsFz9ntxUWPuP/
Gtyj3NP7X0Pzm5Gc/wyVJz9iPEs/+cGV+5Q/AUNbUpS6ALFuNsQyDqF+V0+7Gq1n4V9w9a+
azcgRCiplawjF/m/eJW66dBpx0ECJ2D967/TZlJtUxjzSdim2T8B0/znM+j6mqyaAdcDHL+
pL3NJ5YJqQ7Bo6YWWfrGBUw0S/
F8EwSv9b1iINM6968EHzS5Cc2SaJ1jih7XfcHv2v5QxStDmn9pLbLQ5B5ZhrB9zDDQ533Bn91N1f1wzmlCBw
/9kbqT46jbq8oevl9365VkD+i+ivBV0sxt+0ueJfdXnTtYq/B7jTP7tdfY8PxJ/2vS2Zih+
JbKdhlN7GtEPBP8h0A9B/0Q6N1c9lcXLs4aGqRY1UkqLJL0WcIqEkkCIB+
NS9yG4UaysDwF72GtDW1WFEE0nDKLtAvZ8PXBPvh3M1tIGw0i/
uZoNTUZhyjGPI7c0r1Ibi0hADgbtttqR1FTlVx3wi0mtqUI9D7GR43TSEA3hTrh6WdeT2XHlKvTDS1kmEY82
/v//12nZ3rMtBDJ0mrRM9N+Sb4E9CLQsRpxLC+
j5iAA3pz15jLuQQHsQF4505IKHdMhFJkRMEe0BgaTm0ICwP7GfYJdhzEt1NkEHVP7AC4kx9m1L7LvQTfo1hf
/USRPPs7gHb6dkspfzEMejqyfzaCwkpv20A7Dt0qLCF/g12FCwPb9BWuisByQaZyiue/
bmP0bhxCEFfsKyklgkJVjRMv9N1K0s2uxMTdCynH3QBS5aDy/beS2tqXzbq6z/V4rde//vL

```
        +IF+2wynqHFekH9oCjgSa8dWqIu1dpwSeXlXqNCSwXpb8c25h7Vxekn+9YjpNLrNKeK9/
        y5d4Q96MJxsZrfV8/bx6fLVaffT6fLq8TdPv/zodJ03wDnnsETHFP3TWph0oxsV/
        aBYT1KPz7fn7ela70eHMgVT7Han/iZ13aTVLoAHJ8tp9LPFC+/3
        UagQMfuwm1eFhFdCmuAdcwA5xA56UnlPbhkG3e4TtsnQG4W42Y3Ea7qFEMxjak7esKMeHGuV3pAfbXdb1hUV
        +jKZlgAgZYx3e+
        KEjq3pB73kuPYrs3vdDxl6oVBtFwAx413t8X9oN7h70IyRH9tvUlMvhy4rz0oY87coUaFTuSAedAFLhiu
        /
        r08mSlaIzr0ucDIC8tM2JCUa3cob4jeK53oZ1YBlyfSs0W3YFU40Medv0mWHAr11G3chKCwH
        +Yb4F105nD/N8m8JbMk851Eg9U419RUzAj4yDPqdah6K4rB/2
        rjp2BVm1DsJwsQV3VJX8qNxU35BV/
        X9IiLLql5lDjr2nHofQdrRcatW9Eu7XxgGPRVwT2kz2s30fTt2dYIooGH3FfWgJwzryzgoAyDS8uVxfceOfR
        +bjK7pP1TRJzHtSw0/
        mlceUZuVFzdyFJ0mr3TqFtGf39zbR1GHbvyWEtDb9CUkoIcEgkRX1GqzS/
        w4wdEztsifCMrlWHIXKzcd8byuZx0KeRHfpaLw8hcF6IExW/FbBXE0BeUk/
        Qkkn8QDfWuA469TWwI3Zj7kMhoCtSVSQH+9ihGysU0pXTXt4k4IjM6Aq6IraY0IhjP6hLx+
        iUpn38moAXZtPoagSt0XxmTmL0LUhm6JSuid+20a+
        gqz14ImCk0YKcdtcQCrlFpxT1iRJmNWCn4IuAZR5H30louYeD4EECfwq/
        HJp9n8AbYWdcc5jSbi/BaPVN2qNdLayKPa8leCPgpQ9vgMFY0G7Yz7RvnHxkFHTdgD/
        CynDHhnZzCUip0Kkb7TvHavNaLaA7Ul9B5CsShaMYYksEjwyxL10hG+8
        lvAmhkzAC0AcgF0kZu+D+
        Lla8Ry50A14VMzSVN6QbS1CSZ3RqAgkQVpyZvLorfBDwNnS2BwVwglcwZLth5r7sUy2JHP5tbp0CXyM0lRek
        +JWuTBuWjh16wBPpWUJ6vuSQIQu6As8WGRr6EKfWbGo0ujQoSN++A4u6EwpP3i+y+ZLLpfe
        /gZVyiC4NStJBr3LkY0i9vkdivMRwCJQbSniKd+jS4ED6/
        lUaw6dwCr6VGRq6kG4o4XFcyHxQkb5/Sxaz6Gv/r03m5WXikYp2JW1WKvamltQSw6ewBt/
        SD6atakC6nQRI7NwG/Uj6Bo7CXxin8F0Bvh0UGrqRbichqt0G/YX0DWQQdMTa+
        yWW5kdV7iD6LsgTVf0591bI0+gKZ/479
        IVGUy8Q2VU5bdt6TnppXPhBp9Chw3s0VkJk1wFd0h9JB/0YetC/
        dugCfBLwh0a8P6zYWeTYSi/qPQcfdMQHvzmXABeFxhRE3wS6NE4/xKDf1Tuvc+
        hSglgpNJdDZFmxwX+JQQ/L1F/0hZQAls5WHEBkWTrDf4lBD8ra2yp3IQHEdK0xBp2m/2
        wc7lfQgz+
        R6WSQc90YL7aZRoxBJyrZxqCzCfrYz1YCKQG0sxwRY9DJEheXjVtdY9Dv6NFHzqUAeJhoxBh00qYx6J
        /Z0xPlxHEggHZLvk8um8PcxkAISf//3y2bHQ9kSYBBtkZy/KZqaqZCKonRS+
        todTdFdGcjX3SGYM0dTtXiQctvdEx2b0Wvk6l0zxlCPnWJqBVdedbtGr0hu+7S0w4RINxx+
        qAVXXXCVvSGiL4zQSYIsI/pF63oyjNsRW+I6Cu5z9Z6c6guImipmj7dohVdG+
        IUEKThj1yqjxm0VE2f0w1a0bVBZquyt0D0iVZ05cldusHPEl3n++
        i0t0I9LBzQFa3oqpPaJA+jrTBTFwNJT9ZKZlT7Y+
        pCS9VkHt2jrTCjPrwn5claY48ksI0WqskiusWPiugal3u0ZVSW8Tv0d9G8FV11gpju80PKPWssuoSt0L
        /jydGcaAItVeN3b715P0x0bWfuTt2dMTC4rXkruur407rPD+nUom/
        vtaLegG6mK4ckBoQCWqrGH9B9F0i9hr5lwjf8+
        G6qTo0X0RHYcGmUmsvhAC1V408kvoX2s9lbC0luslztw2Ee+BfCI8IHP7w/+rz0gJ7N+
        X3NW9FVxypkip49Lfqa0wf2bPKyHW/CfvZZeARxUo+0
        pMtq8xxhE5Way2MELZdo1zjZsZ70kUHPIH6CSux4cNh21mE/vRAe/+
        XHie4MAaEeGCTGQ5q3oqu010qQjvm06CN0JVfCHzvJZvhZeHiGLCYNcevLQmKwNughWtFVxzyQPBxgz464gk
        +FJ7/K/y8t38N88CEJwkWGqbGuWMwoR4QAof+BkdoqRrzReLg9p4XPaIP+PV3y+mz8Lvp/
        K0XKH/UWBH2vjbPgcE7/RU60PJTRS8ix+VUck/4bv7kCYRu20v6PEcIXfor9KGlBtHlET8/
        Kq10s+/ND++7mefyW74TJ3erwQlEVZdTGXxC/4D0yVX55p0uSI3oC/FhaaVh0jsui0Ep/
        DXxHgCVfxZV4G1q9BzB90g2nE5soQW0Fn0gMC4R2Qm8FL6zGk0HC8+
        mTxT50wuEkV6ie2GNng0DsUvyKC33Dh0LWk5oPHV/
        Fx6YpfBQYmVhkmzti2k7Pv9FtlqJ7g3r9BwYLMXVPUH8oRfSvzhFbwgtv9A3oh8qHJmIWApvLd3S81kCgHq0
        +mJNSRp8BKz64QiJ93s0TXN92tYIwSn0AaFGEKxI0HF3Viy3q+0hy++8
        ksfFdh8GcAJVvt6oMzJ3mvkWW0We49Eox76xYoA/oidT/Z4Dg7Ut4vh0u9+
        kFpxg2WbufhvI7e6y85pb0Fr+JVrmulde2AwZpBP6PW1fA6BQDHvVRHQJng0DLac/4
        uz4KglTCy4ZTr76dRAVq2QYYCu5BCyJ11SNMbAap+2ZeIIIacGH53XDYPLHgZzH09U6zKz/
        XylEAGtExC9e6nZH47AM5K3k36Nj4QnjFbBaz+
        fnafsbA2FCLQI6p1n9ni0Y8R8FcmdyTIZnx6/LV3Rc+oX3vnrtB6wN5I+
        gYykpdwhYpefZ7jz0N1ABfR0qT3DqplA7CK/0o5Lz2cs4TH389YlfggD52yS0uy+
        dclqvdFmxxiFQHFKg7oQ4CJA452l7AFWQa3AhnVM3gxvIW6J/
        fNzoztcPXxi0gsA3oeVvEMxI
    GraJFU7bR/z3tL2HUAnZjFRH3HPxJXoZynl3vsn89thbC2QWW3CAVTYMg+55
```

ZyqsbsNC9W13WZ4j+PHtUB4t15nVCMmRncCPvxu8mJDZksmrSHRESGwqmfpanjUq7Tmw8gl
/CZ900mZIDoAMTjDThA8Ya+
j2oMwmi7NqREeA5cW0HaqDLdW06NI8Bwbzb6N5d9yYhTYyB0iPFrZrGK43G00z0IFNtD03SBqTSkRHCBa
/R10UQpUclRad0yAA0eB3S3S+zJti0SACWG8xv9x6cBfL8X+2
N0z2Pqf7qHSnBSFxqaSw4CuaeatFqud59PVthebsvSEAJA0DruDc3s1fzYZl8gwlprpX02dgSSVGr
+q3fq+w6BI9BwZjg64pguYMfID+
i0vfw71DJ2jQaf990dVKdb84DoyG8AXNTXbfCXkufhfdeYXm4L95dB/7
vZdCM9g80rTV6I6eGFTybkLFoNI5sINAZnC5zoyemg2JbQAs2XEqeUD2rAE/
eSJRd0FeqngxbX8DBhWDMFQ3B3biy/McYRhdlZttyBwWcXhw6Q+xi46v+
SG7x00nNw0sKnEt6g0D6skjUhSZng0DjkGX8MJshueYbT16Dm+5
tvSVHbcSU93F3MTLaXtNZx9Bl9RkYgmIJrxEt/eN0Jz544VY5cruqm/
qeexg3ksRUSYD1rw1bdex3o66ng0wwac80KsBnjNrUxgfvoq5bh8S39LvcVjvdA81EuPSmEriq7b
/TU+NEzBNfIlujxtwxGT1Rw6dNReUfffW1y0D2F/
QbdRIjGN7g0qiLPCtuiZQKxVFfxdYIIueov0Br31ZVivtza4sF3Q9Gm0CnerPy7zTsnx6xFgvdIkR7Q7HcZh
+
PnU2eBn41wiMkpByl5zIpynAegL4bzScw2LzYNzQXld2YjHM9X09LTIx7ausWgY1d4reao5fC9
/PMt5iY6BuDFE0+5wjpjP7F6endTcEPj9Edy8Vd53q4LjNf5rmKcf7h9q0+
FN7bTbdMbBfKIbV4N6W7hrC2P8J5pnM4t/
q9XVkKpy7KuF7sc0vtrHhcSRTdeiqg7weL2HNcg+gh4V0TREgXpBQFkz98GLxpHs4R/
H1hPB7MxV23p0nfVLiorXkgAWTkyyCcCML926jozqJ7wnPaMRDBnyi1GzdFQJANwlLvcI4Q7mRZfnbdedn3T
/TApTsocYy0yD6V7vNPws+ng27sfRKen/4
csTkH6aXn8tG7pyliRLIp43q0zVR8dPc33VU4Rr8S/jLCHw+T3ewsfKc5B+
lT9rc8R8agRvAMuwYRxECRRhzicb0bKmj6WGrZCYSSyoVfd7YvxWARuaHohj0pwpQp0Ql8hgulEWqHwd6lv4
/ReE1QeCgJ+okpuo2jyo107T1HxC/VZpafpfkw3Lyuk2Q87nR6J1YfbI/
HVWfdZ5U2mZFPzFQL6daCpGFk8Bz4uPDCI0QxKcFBR8+/jtxoZf3wNRn3VsfR8jB9H+
wWszjyHNt2DX6tpB3vq71kK5+RapP3gJM0XIErLYzhoxEeBPEnpAL/
sHee26njQACekdxtTAnN9BpCnfd/u4VsC0yBUCwjjZL97jn3/
uMC9oel0RR7PP9pVe4mb7NGZTwY9TqLYTnyQ0m3kGe5EEtV0XtkloBb160Z6TM9zSt6LL5tR3g5uDQfjZM2e
+3I4HyNv+fnUXmvWZpk283oc0leTgM/lJeP6ucoo+IxuvE+QXNeontD0kcbMF+
ZSUhEYbroDbKSBwdeGbxFqJBhJC15z0JBAQBi0svG7a9luS0vfIr80KpPd0NxlpSX6AnpQ1ZyiS6g5pxqVoa
/LHPBcAzfHucmEuqTCE4jU03uLT53XCprF0jWQJsIgIqh0UF6vtpPsi2xFaET3H3/
IcAd7mkaSX0lV7h15EeTHee+EVuD3ShxSiuMCK9NPq/L3RhANYp00I3So9x1/
zf0DTqykpXkLjol0Hk+heSHcxXrvmlUlKSfKKE2HU7wyyt4JtF72n1jl/zX03I+
nlTMD2pgIpp6W7zsM1WuQNui+D75YydM32IF2MKiVxjBKpgvDUUcTf8hzArZMGMsUJeuazG
/0SH9PFijQyyvnB0Wu2Ju/zsrxh0zl+NFxtZx6oI6BGj/
PXPNeUM721OdWd3SbdDUkjFUDIDXrJdLZd9Z3btvttt4j+kA6ZQVKPe/
0xjLQwUPypZtD4j9EmfUIakSVQxo2brcqoGtKBq7K3hcVhdws81zVsdml1TcsnZTZLd9EhjUhEKATXS6Yfg3
/53kzJD3M1URvk3mCKRfTY3oEHnPX4Eql42TTiY6KHP9tQhGsHozG/
TXP44A0Ube7puVAmDFZu+0YdFIHhIIRrpeUxvP0u1qC/
ATUEbAxILqkXQy8ccuki77dNS0H5ICL6AHpZAAIrwD3urdqq/6
nmotiwu4V0o6kDnfPcUHaiNRqWhbEgB0T0RuklQbkBPGhh7s7zdrl7+
ENth3DWuC51vVwCAoIYNGbf8hkjz4krSQIufDEw52yUWBhA4l0s0iA0RvSiAQF0Hzx2oE0YWF6i7QS5vvQAu
+rQn7NQS0AZCW3M0rtplcE/cZCJJK4nlNS17nBqHTfob6WWUs0a14dMXQS+
LBcI9bEyCHTa5e16SpJc3pa7u5mtaDozMi679gU5Z7p0uc9Jlw3MR7qM+TFQj/an50+
I205A0s1YafW2+puVAncFlLZFmmjmv2e5Si/bspuzWtXwuvzG4IW7S1S/0h8qCrGG+
poVH2F3/+YMDWEz8W37+
Vd2UrsluZXZw2uLt0UI3I03UlPIgGNS07ElN78jw4TN087G4dhRKuia7U9+2
YhdegoCRtmhcxHFQ13+vQUT62ShNd2dQ07LHXxu+tAgRaaYNuUkq859k9+eV6ZnsNkbjgvX
/nl9jpHL1BjxENx12FzCgh+Cw5RIC9ky3nUvZpaSvidT/kd2q3Dh/
Asw7TTQjMsEc0N6BTDwGMyEkDj0Ih2PR47CA0vviJ9nLo8Y0FoVuS/ukA7/B3f0nu6+
Zr4HiUbxmPttdQJ1046uaJz5lF+
tBNQrpenxuMJnGHhTFjjQQZqw9R3j6oIpDxJpH8doxMmUM1JzkdJqlpS47AoDXaF/KLv+
VfdFAm3Y3ToW755lDhogUR0dRvLanrJjYbN+4mjGcob6KT7LlMHIunuv+rgXnMD+
hkTXmnrsDMoYPaM/h8U9EBjvBCqhJegIG/fivruJhWlsuHDrDqX/9
L5aMa5Fj5p53d2Sw77WCItgnHoQf5jbpAiLSjiw6Ki7E4RUrPn0jqxmAQCgKd0Ev5p2158ZDWoiQD4SYRZWq
+UVvsqhSNVvWIjAl/YywWM0RYN2T9E3QTuDaV8o4u4pBZdMdxIJMMuW87WJ/
vmbmgU6Ngj8EtNr+
af8YrlqXmqunzAT0QpbG6x0e6QTwNBzC0ggzHlWqexbwHHaH3CXFxWrefU9PmsvdBEBg4fe5N6TX0RP8RX9L
/oK52sWzqqJRJGrdrc2PGouiaoVAShe8VXN6WV0X06eH5c05sgUjkZ5VKnuiVpPX2hLy1M/

WRUl4sHorH52cp6OY3jNNkjAO72Khcfec/ODCreAVj3NfkpxFqAfAZOQHoTFj/
Plqn09d06a+4
Pppeb8u8z02TeCPOZbmGQDaPPktWN0lCHRd6QfSUlRmr8tg5PmcllS2ZzfP429CEb9knYEjJKAbzcZtHny2h
/pvkwi6+MdbBgkx6WrPAcEHZklA4gwzCqFQfpaCQUJ+
mjCM0bVXnSPKh1f9Kcd9hZyjVYgliQUaqAtlep7lkoeG7d+WJSQIJMJzxpLg8x0A09n90+
EXH70buNfa03fqQPaHuV6pOBV3vSt6+
Teqqb82Tl08nzXkvAz3DepEuqgCUglCLKhfGCdIQqsSHVH5FB8MpkgpFQW7XjJjrTfDHz4B5M04UljcEWjB
+jKxSku1yqVPcE+uevCchC0o+kiZLmUDkPtacfMegCoRQU+00MrAjDHYt6yCwOiLxJ+
lyqVPc42cMfw/r64kRlcz4bypMo4TYBjSD0C/V8xbr+nFsnVZlb9CmbKtU92wc/
hgUJIHcYepAPgdDtOCdPaNSFe/D9aZTUicEejJ+
uEQm0vkr10bYDv2DTJWkj8q7a3VFIJzpvD78Qw8ZxkoYJ2ANCbDyiFaP1VaqPZsz8grWYpFLeVfs2oBP9tQv
/9jDpCLqxru7HzIAfTxUVKWTqaCADQ6bqAtqRCkBOwCgEbMk3J0rK7/1
KNQavqAsakgN4UEXd5/cVkfzCDPQh3YTaCLQO7QAZTjdb2V6nuKU/1
io5mMv1lvgWYOxmldB1ZfW+BLhBSVvMrdIGQGI/FUQPQ+
ipV7TMcEJIhmaAfQ068j1VA13HqtXs13az6SbXtSZRhFLnOAK2vUiXyJw9+
DPb9Uu4kgwmFls6QVHY+
XcefVxL88cVZHbDNPbAMARUyTgXQePaDOs5jw1qYF1ffF30GeTkG2ZvbeijpKtGqEQsBrwMhCUiVateOCvQz
/xZdUs8raoIqrtvlUF513ckRttEWwg3X9nnOIV0md3dI5FSO/lg07C/
YomdQDPg1Gn2ZXsguKX1dYBPh7R/yzrQ5bRgIw7tafIA5whEI900Evb//7
tiqBOaEtuoWFq5z/RDZ9rplLoPK8nadwP+F2jonucytuj8pPlUPSlDk2Mo5+
wtx5sOK907OqfU+XxuUvsme8+DvxEy2mbnoOcytui8AHSpeSst5NIYCG82nlxqw5b+
Kh76nUMt+IqQe4IiN+kLKvOItevUJax+3zRF7wtqRz9ygBy4ff+BuAMxhcg+WPZ+
y04PBYqO0KiwNj0HRrJcvRAooS1kAuh+lypxz4efKM+TG0AhoDrLvnl/
rgUFz7dSUGNdWn0XF+6
A1tNlThwAXb3rc0nbYDykgg4bJvmMRZHI7q3e3j1IwWL6RLBx0nMANRGwRdcMdlewkrAc+
WJq9LL7q4UnR7yDYkGlDKQ+7yvaB+4OZUfJW/z2AN0JWBGRA4swbrMNNgY+
IioFhaIgdGjs1h1AGLIE2qD0Xg1KakdnDjcOdlvexnwMJUC3sWXiqOdHXiWs3LVF/5Dwl7+
lq8XxLTrxO5QBhBXx7cydPHA/3xGSkaLaAuVoZMZNXSOl+IreO/pf/
R5r97DrakGXsnKPRXe/HT3r7W8ZtujzPhQGxqgYSMHe2p0e3TyIi1EyVu7cBOV+O/
qRjjnR91U2DvHs/m6rE8bj9vbEN7Jz1nOErogzd+YaKPfb0U22r6Glt+grwHu4fU1uL+
oPNo/
DznI2S2losbd237q6bo8ZyijoseiuT0fP9fbF9S36fHxXt9Wn24vJodeeNsOgwswVjZcXhReJlqsHcTG
+lIoYagmCkqajHyFjfaoI/Tnnwv6ZOyJcgP64G7u9e3r9dJv4Tw4G/
hURGsQ3UHX2IO70WaW8hg5AgQbCcidIpwtP/2qXeUhv5e7Xu43Vye3t6Pm62xT/
SH6eMsPO2trdxRb0L3ZCVu5cBeXQoBL7HYwIHRuP7rkOtzObt1tHt6vXYqOu0jSSx4aw4PwsnPa8L
+QoTlN0hEhS7sQpZQkMofL2otu/574KMuVOfoliTJ164ZpzM1fu7s+
PPEgp6Lqij2W1ozMfFJhhbeOTV/Vuy3SbTD+azX9j6DKCgmbuD+5e5usFEgY3JBAoh+/1
JRCPfDBA8hbYMNqfrlu7MJ1O8DcobE6fR5PFbNl59MEImH+g1aPTnsOjmIIei+5
qruUf9CIww4eBZ5cdyq1f06kaNue97duuyica3f64HvmeSaEwIs7Fk9ueRxIypL5EL0HuhLnkCc9GikDYANT
+XvaNQXd9Frt/Fn0OBtDNEJ67fA53ZCWnoMeiOz4d/
czczPiwvHsWOSeNn6bXGnDJjr4m9ZhnyDmoON6YK2Fuw6XoJcidMDVPFWHP+ZFxkf/
SdPU9d5k8sAFihbN5AbeRVNCZUJUgd4K42TVTmpYWHl4r/
mx3MD3cJKYjDKocMwINDM20egO38eUcueuL3hH0XWVycLJ/MCZ6dva+vukKFkmrjCUG2U/
Uuampkgu6ruhLFoYZ0W9pOhQUqvFperAHdZm6XFVgiyanQjwAt/
FljTghVCUImGEOTASwIOyZ2DTPOT9/tP6U6WJIIcKqwjGvYI1lZg+64wh6h/5
b9BIEzGSI7naPwj06cPvzL9Pxq8t4ANaIOJW5q1ETMnfo2qIvWBhVI3eoohHnQt6mZJyYXlkBoqqd
/2gF1sARp0BdcJy1rIKuK/
obC6NiRPRuwMbZ3ueDxW30ienJS5N3BHvsU8fMOX0j7ghKO8UiH3VEFzFl5hJ6+
dkHpw9S7xR7iRemvyRpAmOwiAr5R9qun7iDJyuZhZkiDT9QXJIUk4m8Z3w3L/
q0Dgj3NZ06NQH7YNyVeOEOvrT2Ti3RQViSVHqLpdtNCgtQRYVgdcAqdb4O8dLxO+5
HfGldX3qio6RbvOZE71Y4P+JmTX2b604cgVXUiK8zjcB5SiK6J6j/Lv0+u0tb9BHg/cz60/
SRB1b58VLCBtxHXNaajugI/n8punoyLnol/liFmE78C0bJrnrECw/
cR0mbcEIRuh8ZZ0j0uvlPPfUBoSDTrS+Q8WqIR9CHMiBlcEMCeTqi1+
WJbmBUS5c4DyKP4q6YvrP+rrp+LZ/2
AUqBuAszoHTOb8W9PPhZdJdvL4cDQCjOdAV28f4aekPctr7QuA+
RsMSGSjlEN1HRPfOzmEYpH0rf9OdEqplt0wdM/I2V2zFxYu+6
h6DcD4E9JZ8XLnrUYsNUtW8BbR7HAIDXTfd7Ukz/
bgPxWxlO4mKUsL6vlpboA3miFz9lcc2mmWsOxlMPxPOHNQBkmm7ZqxcmvqTSKElBB3iRtUkfAZYh7Tmtojt7
+aLuRdVhT++RSfyqI5QF+W6IuyiF54RfcmbJimj6DDLBnBVFkqzDL9VYEpst+

```
    wBYPSFHSIZCWid8oieuFZ/5Hxz/wK/5b+Q+211756j3mNwCL9yp/
    f1uVBVNgzc0MvUUnWt5UR0btMbJRgDajh+ZD4TC15UZ5heh/BCNm3DWvd8hR0UO+
    S1u7kYUlEnwHCDzg6GHOk9+YzeTTtBqT8AeozLacJCizSuCzoJfJcWApsCFqir6QNaim+
    ontbNkt1o1XQz/oSb/10
    gRG8A8dYtsvffvoQlmiHLu007gBYiolMzAsolsj0R56D0nkylbPnB5X1ZDF2jJnqluVKCh
    +5
    PlLxO3VJB1kdTdGF3e9juiq6y1v0ylBL9PZnM0xChulT23ZFbT4TbKBUovsHlkMfoBSiF17RsWNY9KbWDn0f
    +s2zVk+j1K0i2UysrKkrN2DwChFDMWUyq6m2F/tNQq6JPk9lAaiF+
    mv9mPYRy3OIaGoMAVUCnIZCNH9K2m6ENZMxaZqeihI36VjRKiTkH3mxwTROm/
    DRTCGRFunRdL7cj62uImy73HB1/
    anem0wbTlEP2H2ZvObtHpCZTuSovS89oRGn0AJcHwEwjrGh95F/
    Glk9Py9cehVQ0y1iB1MT2e5JVFdCpYdDQ8Ljnw9QfoEK8znt5025CkOuyYONw7UNDPljdm7dpJgG266P6Ehf
    /iNl8GBLd67FJSPP2a49jWn5GBQ04HK1ACgiDkHkE0vlt+
    dM0ID7T7ANCCh9SNulDOFKK8ehcsOh+hU1S1UvbV10OefUyHl/826rtFzkVdMa8E75yP1k+
    eGoFxF/sQLlwGkd+aUQvuqIPzD6yiV5/6rjJzMQf+Qp/aP/A/fPUoBqKfomOCgH670kt/6
    QHCCmshexwD1Aa0Zm3KWq4NjKP6nq+dGsn0V/
    yjcgkKaIfmbTkan6yPPqYB8TfyDpYqE9ZBA1t0T9YGMRbD4pD9dgkB1D/
    Ivpj1uP7xd6ZdqfKxAA4mbCIYFUU3Oqu1dpq/v+/e1+
    tVG8rqJSRqDwf7tJ7zj2WzkNmySQD4v8xxIiOEIyFBvSd5Vatbl0KKr3mSsTcvaxSiy6tJyyzXtFNg28I9eB
    /X4ToY3iciB4j+
    n2eokf3U9Ou0T8VJKGgLWuNLpO95XWHOB6jBkr6bhy58DgRnfSu0cd8Q2gKKTH7vGWTbLCCDW
    /pi4yiQsCvGbtBZ7dNMSlASEgVX2F60d9YHBsTfnKnS/
    SoXmfqc3TbPXu8tuVV9nlWjkSW83nKiXfovfyLux/
    Sp9KVJRMGcdcCbSiDb0gIEWkz42Zn81O2rGVuf0lAdRbEl/
    EGSlTPj190TXioiH5a9Dus6H6oBZL6VuGreX6JTtyAglOooGnwxTQtUfXHfkLcgkL0S/
    E5GRkBHRCs6gX/h6oWe3GxqI+5wdfgBIAQAwrYjdu9iB4nYYa0ir7hm0HvkB4Fr3xu+
    a1gTLxlVCzRf4CqsnKy7YzQsDlX6JAs8xjlnpkXHiDoAW2+
    AUSlzboFR6SvMMMfCb3tq9GUruAINBujVKO6mlSV21twGoRMdFFcq0gmnaJ7rBkiZ9g0rGjDNz0K9gPL9uA0
    +vnCKUk6SlcjJr4OMSv0r9wqloZG0TusETLKc7+xz9BA+BsIE2cfZ1w4zRtFO/
    MFX6AVjvvMTKkzOKTuxv1sl/sIfZ05rk/0d9YDGXZ9H8hVjORp67ozV3twCt/
    gHZsioH9hhf4wsjwdCRfvED44T6run8fTOwtjGGoSHaHKmUOGU56v94FcYYYftlf+
    rp6s4uI5MxVna3vLFxxpnpq3hJHXKHN+GK0MbhBL247TJzpSxpI79rC9dL8CuULIFAU1g/
    eMeurff2s0D+vKAivsNA+Sp4YSJ5Nuk/NjmcV4mrMs+rpEhxZnh2GXu+
    MKAgBGkmeMCaPvEWgMGi7sQbc1omhUr54+
    zd3sBXOKNP8jFCQMvTYT50QNfvAQpd2rPztxikuXcUrVVW2qIZAnvYMX41kj7IXT2bh+
    GNULS+
    jt71vhfYxsZibtPQUwv904qkEWIAg7YdMlOsJrFpKX66OlFye5NtNp94tdspm58HyP1RpXs7H8fLtnhI
    +cPC9N4JjHudjyok30F/4b9suiHVgaJE+43GJwBFH0W8T8qT1Xuyk7ZatfDVDW+
    ZTR7YHKaPyHkppFMpcrgBAhZi/O0Sd58o5c7PCyx/
    DMnsMsJpjryr92u3xzjGENIs8zObGVhC7RocEpMcqLmOm6dhR4qxKfwOm2AJ/
    Yc4SADc4cSqhhO2Dim1Jq+ggKM4waQorZfqGpXjBCjVNApXrCxpt+
    FMBsXj4OOltK3aWMbmu5gVDRoF1CV0uEzm1EJ97iVLufFQBUWT4yS1R2nK1L9BFfi92fjys5SR6BCqA1WJSN
    +WBsd4lXD+Nw4g/USSW68NNudECLNMO0BpE
  bvbH9oEn1x3fMuN6NkmNwkP6gOFb/92l00m/
    ORP8NCc33Z50F8A6wq6ySSvDuqNf4ddg86d3cmWkRXYF/+vI3+67
    oiQfJDHdMt0YcR8aFyBmHG2UMJKUjWnIk1QEzfE8hBp4Haxh2CK2nf3Qj0RHRFfBl2fdQJ5Vi
    +5ZB/pyR9qHyZvrAGjPgikZ9MmhRnu9od+
    ZMQ9A47BQuWAy0BQQMNuiSWl7qfExcAsPBJOlrOu4j9hN24DCHi/
    Vx98fpZm0xNvZJHOTOSKkp1dIiOsLzA8s16ZuUQyp/6
    oCwFWmtB1L3YtYKd6XqcyvWt4i3ve32GoBeEis1yqIHS8T2+nXnwpblfMQFA5WDdk5+
    VpeRNh+jEs9jMsmFmltPw3Q9anoId6lZjToGkAzZNoq+
    SHrz9bTnkgHLdYoPtenw9ordNOI35ypSJ5vW3SXgI4wjnecjkOP8guv59CNrlE/
    sVK88JtLkYPnsCjBjRme0QToLZTCFoFfTgf1CX4sk92MQs0onXekQvx2g+HM/
    cfJfJWGM25kWpGBmixzaxVdDJIJyvZiZcd0nhQRfpxGM9ojsnp+yjSS/3zTB39/
    DLg6L623WMmVgHTVdbNab+h5mT5ZEGYk7SiT91PAmF9FvzRW2qIPc7YLjfPe5DsUy/
    irauBJaWrnJSAzfn0abkVJ8gboMG0KMfmhujmZu/5YdbdUatWKWfQ3+
    mWlJfevzjq8UIVN7DTcGIhUA8Ah206B/N7beGKUJzwDrvaBaeX0dvyJooeXGL9D95Pst/
    vCnoyEl3X2nJl/mgI81Lfk+KVdjhHXYj91FwVyC0bNYDcRB3r6XP6VlC/
    mDuTeSOG2aYOr7BJXFEyXflOGXavIVGRUC/tpcNE2sgoSsGwh/
```

apM9FvMgVyCnvrqOdKh5mXcZakOYAbd7xUmguZdOdmTjMfOJruCABBX2Wwa7LYuYgBMQ7Rp4kpXBKxU6cpMI
+RL02nJO4xJIfq1BEysC2IfTw8jb8ipIE+
E54CCtt3LLQ0vP3QNrvdkPOxvUNlRBBkXV1Wvw1wxsT5sKyYg+
pROdCFvcoSZGNFLeuo9NyYyHvW/CRjfvMyEDIU7YWqzVvzYfj93LnpDTAs250PGckY/
PeIjNt6zfN9yS0OeDekIVv2+Re+JEZ2C5xjwv27OGmMZg+Ee8IasmTVAhst06oEQXDGic+
cpRI9O/I4pf0BB7ltxO8ixYvsgdO93112U6DUpz0QrGOUoVQ0+sBHz5heNW2ftjDD2J+
fbfC3vUga1GNGJ11KeiU4UDIi32GpaP8rDN8bFtfTzLJlYO74V+7
NTftUgYmYiMhzHMIjPUJaxSEcI5Yj+CY8PgmcctneXJToaEpMneM/9Da/Mt2DR8eKr9rrB2
+h11V4HU8tOW+
uFwYmQjApCCC05orfh8VHQ5B1VUKAA287R5brNFArOlZzQDzE7NTOpeU6kPQJAo86JvIoI6QiBlHN04pWEJ6
/k7vXlQEMuEiW8CccJgRLUD8evPAO93cKtFUgosdx9/largJXrP4/4
LsHw5Ur0cmFBwmrDEN+TS+8MKzC6T9JAuquBz/fHDGZqD3Wh1LHXomYoD+
yiUdFtQcAqryTclAJVJ20cSkPuIYEopPLG71fIM2+7hxvk30VIhhF3joDp9hlDwC/
XOxLekZGI2bR9LXt6mo4K2nJk7lypPILoCCIZ9wJ9f6/PR/
L1j5V9mTBoDJr4tq0v9DF84kb6bq+mIABMpW3FbjIdPdt/6iwrAw19vXPwsH8/
fZ1BwAEF9MvGNcRqAl12F5jP0w5xe3PuuMJ4vyXPmB092R2j7CvBkuzWFEL4ez9/
bjSKoRyBYbSa+OQtQGU02SoG6bYM/xO+DQK/iS2qPzkwPnuyuwGdeTQEQMGZOX+cDZd8tVI
+yuuY5eH5xXZCKwed5nykA0NxbEbcohdFfvdak1m4K2oZ7hhxYBQ2biavLOH1RgRq/
HM3fmwEUbFX7GDJxHgzPi47gXfbpnPfl9PgkHiNSWx2hdiAcYYWVoPa2WpSJtwjaiPsf4gE8LghQ30
/KY++fK4Tw/ShVjkZPf9SGCsy1wzlh1MBElQSA12S6sG1yf9cc3YqfbF9C/
KRAWd609RHU1oNVd1gyeAfJknyfGvfAySL/sXenzYkCQQCGe2a4RRDvA1G8Il79///
dRjaDJB4xaKIy/dR+2MpuZSupvNswwMAOFziCCZxNHWDSV+7
J3UuZw2yHjzOE70yCqzqXY9X2gn44juJ6r7WudqYDt2LoOi/ynTEqbnvaqM7WrV6tHm+643
DXDHzH1FASTzbJD5pPcbfgr2Cg+9n3317owM9PsPoQP/
hKbyi3z7wdmfhAoj9Oxt1UtInefy3iVD1Vq8Wh+ZOaBGaEbTqePwre3vr93XwehqskSf+
p6N0ilrb1bfxhEb3rdsdJEobzXb/5FoyGnmOZtsDM007xz4Zlvl0Er98w068COz/
U210TUypfZEsvQ8bP8toB6eagxO9MWiGeeIAfsTqlPlKtRAIlZwvALyw/zREFzuHh904+
hMx8McKHE0L++pVVLSF/5
f8ZgefIv4Xi2VbYfkTMyhw6B2j5mAnbZzNmDPTaCLUn2RP8b8nMB9FTT3NSnCj5HTOMQTvBjN
+D86kDDBb1Un83LpuOfSQlJUq/axwH3rNQ0hIDdDjlOd7g/
jB8Mnee7uovoQvp12MAjTlmRmvgjEr/yqgP08jF0y8fk2KhhyW+
kJ71a9S1bFiZEai74HYWNwad1qZvIdVeUm/l33oCGEA1wEyzAbrCw/uCyrTaGwf2a1wZJj/
iPMeWmb/O3RzOQK06DfWz9Pfa66uhRrWXi6bC1hN7bO1kpYt5Bcp/
ynID3Z1Wt6FHsZeGwIkioQMMEsx4LeCqfN1FGZX2bLFzqPYyEFgHZehL+zDUx4w037/
H3mufRE2Tan9xAscq/bi3cw84DjtU+nX4e+3
LcaA94YPWhJ5fO8moH35UtZjW5H5Su9GorejmuZflKfZSoo5/GErNh+8K/
GK4brSo9dckqqAWo4sZcwlKP39eAIOIjt5fkECVtkljadZrCzMrXZmrDncrfftcOxmT60JfgFLc9Tbx8cBR7
+/t5K8HKfMu0mlR+uDSbwamXjWqNzfgbvjdJr+ggSW89iVMc6NTq3b9wR+R1uCKpih6/
Jd40UxWNJZ+
ssR2INy4VzXB5M4HJn4LSGEFST1tTJH723fGQb9MIniWqs6rRjGPnz24xfg0ER/OQLHZXm+
g3HdqFRr3WY2xMWFwjXb20W9hlpr7gx6AnOE6QXzcbysDt6b1/
m1nyRA8nqGZXgkveK2W4tsiAsUlwq3gmQ7ceX85ypdSufQzfZCfYcZYQXhprYPXmfwjZgG+
isSJXgkXQ/
F94lrtunvol5Hhz2mWOJZ6Ts8kkveHIXxZOoaOpyhDxIkL0hgKZaiujZeKtwJVttJG1ROPCt9hKeJQ
/BmkNSq7YrB4DPDnSU0zl+TwORZTtJ1DsXNHDwibNPaD/
GqASmudOIfuOHgRTJ3bRgultXGtD1wXXfQnnYm29CiB9hel/McJ+
nGelm5ZVDpcw0PbMd7Sw5DnBLPadgoXa49ZVue7/ueRbtLvTiBHXiotL/2MjBbwG/6
LDULM83pYYjTZjJfTGSt1+ZOu8OWgMD4USHIzI3qxkEhOy+IcRg0RX4Hd6DET2JQx+
vRfhOlERjwIIylw3wnEEWtaOcMgMl3rS3M7CdTi59l7eHpMOgiUdAUHkAO84WPe5vinXemwOT
/G9XRYQb1O0q/fuWiEIlqBNbgj8lhPu3Ntf8LPEn6gUIGll+
VQXPQx1r2dVk1Kv0M1qTDcfU0DfhbjKfL7JEvT//6
euEi9QDRXJ56VbJAXLUp9dOMEZWuGoFt+DNymHfqfU1mjkHh3ds4rPCd2BqH/0
PcUKDkL6n001yPSleMwNoftSCHudtLvNxarjct3vkC/6tnPfPchTaBWrdCpZ/UsKh01Qx1+
BOMMwB9HQUilzlaneKd9+TnmAE7fBQa/
cPnf5vQUD9pZlLpavmje2Z4esge982s8pS9Lt75zMaUNgH+6Q/02M6+OHPBaV/
nU1q0gYRqVr8eAucAMKiH3v/4DrRW8QtrAyc7cOeQxwFmQf5CG+3rfEqNJrpiRBt+
jazc6K2GAr8qfqMMMF22HAE/
XgqoRAIlZOuvZTnCAGIkKhG4gt8hT8xZazyy8YR64QB5dtfHLjeuWf5Cm4eSCOm1LKdK3yBRicAGg
/uTt5tPuoGJJy2KXz+

HSC4lGpDDcr9pJ5jxe8Ao9aPSu7Qgp5aRLOTula9l5SdEUJSenWCaU2CHDxufhrreM1GyE52G
+nHpCZWulsV9S2cfs3x8aTfWMYOCOKxt3BNilh/
iVWcDh5xZeqEtM5pR6Uels5BKV4oM5k6V72Ncrnzzd9b6OUy94+2qGQw8xFV+
cDMwtoeN5MwNXVI/Kl2fU+lKsQb3iYCnlbu1nWPjRXP9hp/O4KPzDeSjDlAgvg0+
lQ7VEWaaDSr96HvZp9KV4lWA32WUQ3UTWBp+o28A3LjgLjDk+Q+
vMCWfZpPcCCXboM6PSjfoUTa1eAPgt0SeVl7phY4pCjwyV2TBfVSBnC3uif1HGeTxtYUi/
bMaDfRTpQdUulKsCXB2Q+R83R2aWoFtbQq+E8iZQk4L/3
NOHJ67Ie7taDXupAq9fEUtWheAs0Lr62y2CUxN4HWCyg2dT0w50Hku6Y6GKbN1amrrSw3Rot2lzjCodMX46x
+kzj7OyfVJdyQj/+3
OgUOCH4TvZlG7FqZEDAxOaQc4owN3Kp38J4IZAPtuF3QmG4dGPfQ0ISMv1nnxvQ1Nue6W3fc
+hnOMcr4h+k6MERK1CC92IY1970tke3Lid2rJ0JaN/0
nnUk3DD6IHLLfgjn0gVDq5uvVx60KNxv7txI6Qjf9h51LVQqkLDNg2u++
djs6pdPIjmt9cRdvlujMduO7AnTZmrVrcnQeOhtLfdy65PkpNDhOUy/DUOZVOnsF9OgfG+7
kFRIEp++smFnTHK5VOHuKtAvfBYSzwMxF/7ZxD+
FYBciXGae2dPNsW8hzqGn6yOu48FuhMgVyJ8zck5GY7A+6
HQ9XEnObXzhlMNETU1nT8fi0OTSTkRv/Yu9PexJEgDMBvVdsGg7nDfYQj3JPU//93
u7HdxgZndwMdaZPU82lGjGIJ5Z3qw10980G4BzFKEKKWZJpM1x9vqvbOKfUfMSai1EOO9+
Yc9XaEEkT8JNY8QgEhaEniD9R/xXoBo3rMiEG40y5s0L8tyVUaxZxno9AZ1H/
n46hH2dT9FgDhXhuRhf/BklwoKW9ceMRREh2don+
Kj5EmXd1rBRDuFmxF5qX3nTNeqmItyP4Lwqs9y6r7a5/
E2OkdLuoupgsiPOAgRrxXH7cY05ZY2yBbcA8lFvagPonRDkWpT/PGYDxkE/
eImWxKijqBJ2I9DwCA0bNlvg/1aYx+
RZT6pEofjMfwSIwYqYwJNwhYehIzUnsBGPV5NmNQd2A0aqLUp9Rc9FM/JzmW07S0qHdDm/
SwjUuJP0HdhbBpiVKfMB/Cx8OmHUk0+6VJP9ds0mXhL9M/7n091XIn0kZy6lP2U/
hwYCUmSa8Z1W+jThg8i9UKk3/ZjDTndyP4+pKc+s+2
dTAeR2h4WcnuvJQlPXiTPCOVIdSDF7Mp9Z+MGAwngq2YbB6+80uSjpERIxmjC+6
PX6vsacd39e/MCmA40hYjWVGf3L49Q0C3IkZEF9zdJb1d0aSrfxOOQQQ3CL2a5NS6ZUk/18
Q6kU7QHST93NSkq39WW4Ph0FKMWEZkNgAR8gjY2NsXn7RNpBs97S+l/
lFn6DTnhL4UtdoA3yzJbeOct6aacycI9TdR6kPbKRgu2a10T+
xGm3cMbrtH8cKIVBqac0cIvBQdvqsPLAIwHNslv2+
zpljPa4CQx0DbkwOUKwS86uK7KmW6AMEtwksYF/
JFbyJWZQQw8gg4jKGcJr1f1aSra0aqttC65b+JEZFmxLvLjvm+
ASZt5f6VSJfk1C0jnSHux4wPELoSa8e7PlZ1B732vBTBIk74PjMR7lDfilI5Rk4RCHfiOOqEMoRhM37ALACil
/dElO5a38R3LePZ0BMKHUUYyIhA0w/
LEnVq0N1hH7Na4HoPSgwGoxOm2fOvNms1Z7A92TdBxCTbpKGAn7D+7
atrznXb28rDPGJn7I6/W80Zv5WtSLGOvwmNR0xiq/bG7Ad07U9Yi6ihmZ9/
AQxtkTU5n0y8o62a30fYR3wUIyzbUW9TxCoxL30GOA0fbkwvRwp0DfnVF2ev4gxjIu0bXR4LasE9JovyD
+gM+XMxfhUtfkLgibmvztqQ5OQm+MWK8g3Id2opS8+
nBgLu9M2Blfl3XCuiIiRkaMRPQmmXlPk54iTFtZt0xG1kMvifsbGPdaa9fIX85I5eyoL2Fof2TlOASIYdluc
/23mS0ioYd4t0R/1XM7IfwAnO3xTitbp+rqwzVsnDDsj0soOU1Y2+
LZN16chU0nk0iP0gGI4X205AuB9r35lfbeHDEcJccry3F4CouJW+DZAJjmnOe5rzhL+
UHG8NwgUx4zFjvcjl1/
IOcIYwDCXP1Ha2HyRhdrtyTH1Pc57Df6TAjOHWQPfZfqn5Bg4xFkaKzH4NINspMtJFXtTRnF90xUhRF04R6d
+eq4jz2/rCXTrYHP+tG5KzMiO4RLp8P0XCsdwjLDxpMTzwU/32
YOOoEqdbc7ng6zZvXH9fzFhqsP3X+bZwVp37zUCFW/uNVImPE376Va67qSVGnqSqA3BqD/
ZpM8COEVY6pvvv4gZ0eM5HzblhAKGrUWnmpGCZroZHEHdGlQkUT2DQfC32fZnBKdYX575Rap9MB7VD8XIAgU
/x2eIgc578o7rwvCBeFVEjvAX7U8ufYWQBXkXnX1uvDxjhgjm/
Raz3XSsdI1uV8gXIEYDwpG6eJR5Xw1Te9IzOuBgeGsaqTADKCKgieb8x18pBgrI4lwCLcYvZYW9Z
/OyX3I0zcxEqscQMjhyJNYC8QMULuTH8Ib6UIV8ExSE/jI+OiGkliR86RjqUX9R/
OWBMajhi2b81YPXN4izizBSdYHy6YnmZaO3QtoJOayEpfj41CRdyM4R4R1TdSP1eyDCY9aZ
+fJJ3Uwigh7iXlnsG0beXiyQ3gjei1ywUpM4ZVkguUnl9LN8BUY/kyL+g/
lzQL4eFjbSKp0eMD1UGK1AJSdx4gW8zB9CUS30jOE9j901vcxnMsEFjlOOsZa1H+
kZhvEeFSwsCXIW4EIJdqS2ILz3aDP23hzvaJjd4txkCt/CAyLMT0FSA0cJ50Y9a0W9R/
H20bwHbxDuc3OkY/
BhFITiZk2uHDMsv66r4r0oWKMczY9f5bUkgtJTxFWrcYXFPW2FvUfptUFGA8bdrJluAZ8WMXSTvWKxKoDcKG
+
tDFbvyVIf1jqS2AQgWse3GJeK1Gey6qE91pv6TeLMpmPCwdTU7YzUo5PyDo1hPIFi2rPsHnaS
/Y2yqkgjXgD+RWNoa8sq0Elf7CAynGBjrOZcfY94GGI/repI6+
uB8zM8DFKQDfLMDo4A05Qm+nCQz8SSIZpJ6vp6OE3ckth+CnM/

```
        Uo6MW9R8hPEZOyjkvJGV2KE4k+97
        VSXM7eK80wNCo32JcButdMEDA015C2eyBrpppp6oHfEFR72vryO/OiHTc/G6Q/1Z+
        lxKDVkZky8jri0mqk6+tJUpQbqiejnoISK+
        bNVJ9KSSxLRnzx3eddGLUF6Gob626CNyM9ohb2e8afGQYwTE5ao6CU5r0kTZzvkXASWLm8gUR0A3TpFfGjBS
        /LiEzOLso5gfKHKZ+noNxU88lWJi47eun1Nek36HI47QTKxb9fSZO+JaQoaErBcw8+3
        CJG0NWdtu/JiDS77KKcM7hYWZ6YECPCsGUfd7Vttk4rRCvSwfsVxs5+
        aW9cLPQvNTFipLVBivEmlpFYrQGGYwz0jkaL+vdjxFv28GjGbNFuF9/jOoKzEpQbUhYt01+
        bkwa9iNC1gdoHuNJriSe1IVI+/mQvwm8lVX0BwzUG9fc6fv9mjMi+
        DyflnHBoiRkC4MsJjB18gIFXzz5uxLgS2W6mbR28f9CMdT/FjWnnPchk03ewyevC/2
        PEXDrLuUZA9Kp9pr4TI1J9rTsq57zw7OlSwlGM3fr1GXzMRg+v9mklg/
        ea3qNYwNkp3yXoNm7RcXz5y6CaH0SN7bnB+RQE5wgYHrWofxtGjKNRO3xsntJ+pMnfny53g
        +WalVb7pY8bibGnWdUFN7IecSMQbhAjxdSRxNzHu3Mt/UpnhK/hr3X8/
        j0Ykf2a3JRzyo4yGnml/HhcmtEmW4abD8sfF82Tz5tTnaYX2K9OzD8uYPg4SSIcINGwSe+
        C8QUIiNpVjfr/nhGptgM35ZzBI2N/bOUFsV5ok960H00ifGBt0tc/
        VB7lrjGegD7OeddkzXMJiZdQYq0vvIR2s9D19/85Iya5A8kBxqYjWZh7uZn3f7/
        aaSGeVPV6lmuE4ElSe58IZRgvFXs3Ey7G6fe/AOGr8PBNi/r/2F/s3Yl22kYUBuD/3
        tGGQOz7DsYYg/F9/7
        drI2kklsF1GwRyM985bRwn6aljfmY0y71K5HNAuAsmjILsP7v1zN0AlajVlyNXy+
        bchE5qQjaqxsgSqg1JjNmwbTkrtK+8
        t57ZqJeUEmmsXdwHg5cqW2tb4wTNRYnm72BYOSZou4adt5sQaCipsA9G5vqkTFg973AX5ndhClRt
        +jbp5eQ3PdwJ42Uh2riPM15W7FkakfEP59guuN+0klRgOOrmYi4xpSKcO4rS+yCFqn3YR/
        USUh9Vwn0QMKpIylm5htX0RKUHMuTcdkP+npHK/x75qpxcJZ1PjXCGMAiKXo7T+
        ns7fy8VJbKfMO6EwXvJBIaqhBO/r4ZCOMd4V0HVnpD5jrUjCefi9CChU5dbVXO9aZw+
        Zw1CwajTFatEuh3GvRA6DTmx93DloPQvEs4Rd0VkZs/Cfcugkt/9
        ZZxgHbCFhyvHZJh9yNFiimZilcRsw7ijpiMnlDRxramndB8g5EifqBnbpH9LP5C8s42hpkzlBZcIO73pRngA
        /bq1Sf+Wqk5RkF/n5dOTMlcIa12y4kF4FIj1ZMGIcE+
        dumh7HflGDTA9KOoSpvkLdORI6tUG/Xu8hfxS2YCyJG+
        CNOcrMgZdGYLuEYrDbXut7an8NuG+jnqkVit4b8lH8u4ar1PKeSlDxlBSam5z/
        l3UTdqu0ckxo6+6yRN61yM64a0+IBSF4NobrM/jrwj35n2m8Y3AiELzFo/
        e5Um81ZLP0FhSzsjm/F/YizqCkZlKIpyYQzdKNtLnoPMmbksXhSHwykb94XTMCXfXD7J+
        IYyjHt47phmko+ueuQBQnUnKH9h99H/lYwhGZpWvjZh9JEE/
        gqF5laT0DKMoxDbqT6D8LYEIBTiIVI5gAsDYS2JWNbcCzS9dDIL88LbN+
        b9COB2bexVJNGGm6wDks33GWBKF7a3rqNtluUdSgY55AWgbdMDnS0VKhoxrH/
        kO3M6RVJdszP8lOsn5S8N8K/Cq/HMY6VBTVnCqAUaBiEDHwJ6VexAVHgEu9p37+
        qrk7qutOP/VEbuvdg+
        uvkiwqIFglO5sjj1orr5nOEDBiIF23Ub9AVS9DTBDK2BUYYLGWKVbuk7ffA31nGMrQf6euSjDOodpEXSbV5C

    Ghm27UVzJmNCoi5N4ma29XupUq4xHi9fRiTMQjllL+xOf8tI0moHQiasYx2oGfujLUu+
        E6MByAG1m+2iVOBKm/r+8ecJ82ufsQ71nApr38wxBXGuiK5hiO091vykzJDgGC0y8rla6zr
        +m0Ij0EMRJ92Xa4g/ufg3jEnoL/1RUQpJUqJBDsGmbOspGdKelOJ1rUdFX+P+
        ypKYmNzGd98c8M/
        gPVtQZGHd8ogBvrL0D6s350K3ycA051jXm37clYaSuY66dfNQioTU9LnklDvsDn/
        PdwMJHGrgnZ/IerscZyxcSQWuIwHIga87cz040/KmW29e8ccBN4sTmJ+
        a0XHxVgSLYbBVH5xjjbnv68zy4rxfXgAXfZXaOmOdrpKAFFDEms8GDGAUdf040/
        G744AunvMUZtfxNxcjIyoGorcKj1KqL6JSLC2Mf99DG+qJNU96M0Q/
        WOvnnfK0RN3XZH79eZRnOIQMxDt7XbbXdT3UQEL7QRaN/KcqyD0JeW/XD+
        mO5LogQz15erS6Nic3wMDq0rW6Go4wIlomNVwm2c5jyoS86tkrCTLIBSJGahtW3YG/5
        sqre1Ex/y+JvnLRsLP9qazntclMQbfqFQofg1XXPSmNbutdh8EHOp5M45hr+
        MBQLWzWwaivepVT0LLeJ5Jo+V0ggdEHaOpba/+G8LXHRVzCI57+XDu7yPE+
        q30E1fFHRldSbwZEk2wj+f3w+iPTxqi18fvw+F+XE9+
        qiv1sW60nn7uEwaMpkijiaIREzCYt+wpmv/Eac0HKOys65soSbTWADMRc1buZH+
        d5logibkp6Wxzfj8M98M5ifrlh5+1
        LOedQGKVmvkdoy5K1LSDwjED7mhqi059n34jn448gBmFYER+fncij+
        lGYgtDbdeekpha21l6wRjIGqXoMw6SWXpgxAhdc1k/bZ+8
        mMIXFI8YwGDbdcSWiP52yivdVYGDud7+
        NlQec990R3O6uV8e1Ow8vWDEqA0rF4FJQ9uEzrkuF6Cky7jGGKW/PJvgIZgAr7ecic36t2I
        +G/bcoqfDBNewjKNb+PkH0+L6myS6dkgvHAOHqToNjIq/MSeN7zlrzuS84Bph0kj+
        tBPhUYgBvDSngY36P6U8mDb7ejAvEmNTyUboXCcJes+U5YkviQ+4
        sIpFBO69h3JqNt8AnCf5NX3VtGE0lOSd4gOPxARwdBzbKfwXKXfGxwgAMYqmSzleNvEaJO82B5CxKGHCsY
```

/pD8AABqPheBb6flBvTbe9PgC6vGWg5M3FNX2nTcni0QV/iAFUD9uWncIbU65a240nB/
PiEdzZ1WUVTnoih8a2XoS5JEL7mF44HerqyyCKBp2Jm3xGI/
QbX5SWI9QWkiS9B8KjEQGYr0f2cf2USqZl6xoAIjwK46An7x5Sbs006q4Rt0RsI5YHIrr5E7zrgn1f1aFRss
/HwUyd54589UkeJvYNhNKlIYmuT/igUu97udCS2qBpTtnGeXyKAANCvrP/x4/
pjUk7M7Lr8N7qcvGc3n+h0rzwYgGDWS8cJ370T96dyxxJTh1uVYlV6+4XwRASA+4
f5n5z1dMZ+6H0RKWem2z/D0rvLTERZU/QpGJp5Kbe+sevuz8TYJTmW0Yy0uuQ+
LhHxL4RHIQA0ibaLPzLr8Re82EYTKi7l+mTiervvtmat8bIdpZ/SGPuTyftSp/6
rZVqvJSLTGqzn2icvodC8hT5wzttlacR88iHhkWqDdlf9UQ/
s8VfqdNudouPCBIzGYUVJwvHrr73TqBPcuiQ00Uyl0e/
DZcItL47TtCWjnovQaXy1he6N06GzDbpM+WTdXK1WzY0HgBkP5fV7e/8
PybqKs7Tf9T0UjUHb0JFzTjh3wXQ1efcrklLj9ZebfJvI3lV7rrxjg3kfFM30pfaJEwzw6DP0HUcp5VT8
+vvm8VGHW4s+FvJ/D3syYf+
Iai4KR4xmqIxtXlYAI68JfsUJl32A7ZW0smIcv2qR3q9I3sslxYT01FcX3+
etCyY8GHuT0dT/3z6xJ0P5dDfxGA/AqHaV3LB4AZ90+bimKoumC7DdQysl0n0eeiBz6/
rLPVDGS9eRa/6WwHg81xus3tT/LuzJ3YLuqu05eAzG0hRNZT9q/g6cHWxVkjn90H+
N7E3zslomtxwjkHkL9JcuIS84N3TErH4A4xnI9Tbzhfx/wh6vNC7mkecSHoXRrkiushh/
voVyQrVBl61YxKnLGRX0q4CNegktRZmvEwM1X2LBJv8WT2aGkSelPvAM0uyH4Uz+
J2bDjceMB2K0HdEaq358XMb1NvuKZLKkU9aib+
duxkp0KdXaAbD9UcuF0w0uvmFEd98lMQRnFxUD+
crUxRMRu5t564cP6qo133hMeCzGLkv0YsPIeW1fUqoHupi818FU3YZyzt937DJ7uTDaN4vyHtLhuuXmT3G
+ZPxps+My93vvgWSmHp6MeHAc+/Ij+ePVgAmPR+jUJVEZEc5575IKB6CLNvpLEMCbq1W8+
urprwTrBGFduXEuTpfgz0oK00l43lojF7VEey/Hqmt1t/
xZfVrVbLmr4Wl0cM01hEa0voeu00ueLzEnnQpWPwI5F9qzcCVCmCx0za9zvNTRzXJeq0uqHuHcQbdnlBVK46
/QkdIVRj2uzgeXQpMd0q/
wqh50hilH1qJIkGGInNm5Lc3h5tLxXCMN1IZ30v1YaHBGEsqSWBcIrgvUpCrVE0RPil036f0eU8U60UM3tPM
/Jbj40SxR3pZxwPN9Nsv/
Jp2JsJeb3YUaYX1YJJq8iscrgZ08V65YTP8DZtbjymeq6Ey4ykS0JKTivEKlzbrbMmkKUCulLN9RpDruh86y
/EASvpau73h7zWxJb1KDlk3dCihioDRcHu7tWPoQoSJP+
NkiiTl5bUvXqSWHAxJBvvVySyVwpe2QSMTNi/
d5q3woqjqPkgZTjVILWftXrI8alyHiMMXJMb9Hmit8HpAhdiak5+
LwDi60lU0aMY3b2Zbzr1yaDVV1Szgh8UViuUYUZYxCKKsEG223EWdzpZX1cjhuBXyl0hI8v
/
QSN8fLYeyHEuEQZjxGWuuLjF3h8uZZD1dPJu2avtpQWYSk5pf91cVRmJYkNjPTvUZsyDujXcWck3M660d93G
/Ss/g7p/hXvoNHdz5vrFxdZwsuYgWxa3uh/58BFYwKNRpJYlP57bsXoPX+Nxv+
kpnChjb0d1Nuqb7Jk3EKlmtBd1sehWrRubofT7qIeBr9iH+deyTeo0NkV3w+
CsL4YT4erZm9Qo/zrLt0gbli0zcZqM0J/JrEBcnry/mEn6z8BEea0XFF5ZU/CJpRYhNsYK
//2gN5vTsr4xk/Eif06FoNNb9RefQzfp6/j7tti1qhfajQWi1Z3/
Dp9H36s2qPeZjDxcIK5ZAGn+CubvLz0q3w9VuczdzPC9Pq31XQNio5N+o/
A6DXkQngEn7wYvrWkPlh7uKHWCspdBpR05F1m/
EfMLseevmVmVItW43rFCVrzTRWgfJlV6TNutzG219uw1JTEG6wfwYU7ryvJBdM+GBrr9Zo2
/qN0S2Y/qFkH/Y1z7t9YM3+KypntfDD34qqaWnfNICSmutfKVxijyvUMn/TkfWiH9J+
BCd6xW/cdpRw/bM07ACNHn3rmTvgKwWxTF6n3f07Q/1cIQK33mi2/JD/0ddNs/
tRrcV8hHJKGl+8uTvQdiU3t9/anSPqcNY/H5nqSJD9DqLZ0Zbn/p0eLknopn9H/
CJ1267J2gMhn8iilzz/M/
inoUWgI0uLWa8HWHoX70YjZvBWql1zVfz3e2nRE2RH90QjuZh7mKVf5h69nQW+84
CuETWBan0c3kW5kv7M/S7bnZC4KLeMq/o0VLjRrXw4PR+j8xd6dKLdxw2AA/gHuqfu+79
uSbbz/23W6u9yTluTUaV2H30ynqZ06jSKIXBAEro6IEpMtGAAfRemyl/utKYzXE/utrS2F+
yEI++UvBjqBJuW2BcB/fmHrjxG1YVeiBefF5uW9pf+pAQLhJV7mdw8C/c1cEM91G+c/hS6e
+vzWneBeJVLtS2cbhv5W1TbszvnUdAFgfM31GmC8PtNJgPES/7IxCmxl+4/
izn4pGUdozER9sKLPR7B9wL8c94kqx2KRwdRNbt0kNw1bHogxVc98invn+
E9xhALbH04nIT2h7XNZF8a+K0oSQRNUWCGWL9MGbHPgr+
Wu0ihlUqMIVfIKMBWnHzsdMGHcksjtfklVKPLNLy1ZT3pcFtXBJzCa7fzz4aEQ6BSI00vN0L0x
/pX4WAMjh3UyVI3KLft1u4GjxDHsPrFzn8L6uQjD+
0LD0ftUXW1NNL2iVy5Bh93JzT6ufx23K29gZAh1PUeTkEqumHbB0s12d0ln3bXA38P6uQj1ZBJKH89idELJ8
+FQJ/lUsGrpr3H/EW8mtTc0pK+kUjgIeUN4ioZN98ybumBYEKgRXId3eZUfrS02H3Fz+/2
HR3IvXhIxA6c++QIREmqNtjubWruK3i0bi0gEfpB5eKZd9StPwmcdoSbwIzQk4hqwvrJGEP
/M0s6MSZptn3fMAT6zS9N6KvNenW7hf/H6qIkaJQS7+9SHMWBpE2crDk+
G2lLrAMyH8aH8SfBxv7p/HBpe9+
zC8IjRFhIYrnHPgv0lNd805eGmzut9ynbZf2f6YvS0bZ8xrx8Bj5PxmeCCx1+

nQ6IUMSUTugJxvYw7YfLjmM3AOE+hjeTxLmBJNDDC7h0Pfp1KUVha/Fd+sJ/Y4SP7USJ+
MXaJEZXIldG4lrqFTaQmDMpnYEQA2+O6FvK1k9HeJfY6lGkc9ocWMmiDsJIlG4RX3q3zg+
TmhSF7ckcBLIL+y/pxbXKpSVdh2ojWaJPxRNPxr6WDnls6h61RMQM7AdK/2
Ha5fzni1I6EbW4f/Dtot/
Wcb7xQMDIvKKDALhzJSWtZD0im4c3agw9EMwozquFNxSX5ZZETgxm4KpE79zT3b2TjXrcITNe
+KLbgdftxv0PkJug3W7mDsOIynF+
q0lhJoRpRdcIB0kovXC8A4Tm5s2z9ZVGvJmBYcYL3duJkUlTo75LwKWtQ/
oAzvLqSrSwduwdRvV6fzhZhhJT32oEj2Xy9W2hnff+R/2LXUzTJaBH0IGe1Ft+
nORbD9Je0z0w4yTK755GsKq2sgLDiM8ScQrbJ8I8kMgFHT1mtnwM11OSUaHv+6
GTfkl9uwk81u9CwFG0sNsbJdnz6R5UfBxMXIDHgc7xL3d27v6tHb2j2ntQEv+
O2NRcBeEi6gSGiVuTWBdsOmELAiXaOwh5w1A+oEQ2djrqn+
QqKeWEQS0IQ7UGQ2OslMTUDQ8DPRu/GwAgd7+
qRWdDhH1LIn1YJYRbvO2BgetIpPQLCIdQSo6EktFaPhAMYf1R3pTkFeOXcrv7sA+
iQqArXYNtOvhJhu+
y29wMQYyOEz9q2v1iBaEZ6r15hSdKYksXOYxyEJ9QxR1fRFVWc7Wyfwx/
nHFLirIFncBHSbU7ANGDQKe6kr85WSUGE8C46mYIVlW/Va4/
KmyeEoXdvf7o1JZjggF5vZaUOKs5rD8O8daRHGcHSuvXB5LnrzyA7gU66YtTrdIRe2OtS7mM6Ft3Uf
/dkjKFmqk5wE1S7Ubx/n9NUq0L4QPE/dVaieYfh+4f/Er/0chbhaJl5zhkeMRTx3FuRd+
CUcRoS+QVVG1EKNK2Jzom3lFUXFxMKOlIZgKGKXsyOOA+otGudzptO4f6n/
x5aoEuM0cioV5WdKOSsuWF9H7SMIpvlJydzwk5jMmdS5HzzmbWXa4H1+3
hT8OETOTp8uIiWlUKjzSq+
xI52eIE6xP6ncnxvM5WjZsviZYjef7bWMQY6IyVRJZg5LlnPXW/
hPovgWTUcvvz1nwiejSc8U1UXFrkoYiukvMCNnSJrNlAt55GhKKLI4kX8l59KVAfBXpNIm+
g6qwAkWCPAj6spWIw/
kHvW3b3vZdzqxa0utfenunjg3SJx6awqV5GBaravIswdiRyg2V9Av2tUlb1SiBg2lZSosu5qm895VGpy5lxG
/
K0azn5IVdg8LX3KCzZ7NE81FzK0iuBV9efGiW8KR4YTtbJd06xedROvpu43Nd0eKJuBKw0HzO
+9oaETonUSUGChxOowK+r/
dfvM6Nak4N9l0kO5LpHIAyWFcLNOXiD8E5zcCSUbkp3wwWv8yb6NDUE1BaZjVJ4G6H+
iBRPS/hGJaLxwjM+8awzy7JVc27/2vZr3xsCaiDLWnx7lhcE5bIpU74m5S6M6L+Hu9G5+
UrnbmgvUrtpIIh+DCGJZe2/kw0Am7pM6dqdRVUioDoPp+FgWO3zq/z5ZB6EhCycxDybw9w
/9H4/rxfmXKKGmcJaUuyIz0JdWDRJxpYUnfSiS0R+PWr+
Dp2kmSP4wcZuA281UxCawxjtlUz0w2DOwNqXEoWrDYMSKNzsyXSDUBTejX5O1/s6T3W5/
ar7gLUfmSpdRB92R2k03WwAVBYzeMv6M9YbN+
UfPdV60xGEXMwHwTfBDovs4CV2c0F8fCzH1JBFG1HUcAeJMgjQYCSrc4gu80ztetN+
bzRt0jVDXDbKyxHyzfr9f3ZRDeuSK6EiUpv1mqlwlcRtOptnsknRSp2b279Wk6nk99fJATa5gCPRvhAEIO65
+i7bE1GwOZsoNeUbjXcQwRYaiWSRd73tEujtq9l4G7VqttT6ednsPWnm/4ixXOmppt6hlTR
/q5e7LSjQltT4Sr2kcc1JU063nI10XKF2+x+ti/
e8QA2CYEEbKkANiDHT8F1C8XClZ5erkYs5rJZfOwCl58zpzZAgdUSIv+
Abmu2sgOeH7tI6cRpD2axsC4ARQ3/hpDoINl3udIPnZ9Ryxq64/
IozDao0C4yyRtV3SrV9ETDDTvYWPYGTI8yUyRpG+GrMDJU/buhdxDy6hhNz06uwAhMwp+
rjogGDG+1HdfbZZFcXwC+adpYgopfIds1ZIEZ0l5m8BZsq/oOO0y+
YKlE9jqjiIj8W9PQ10a8hse98e5f9NnaTr2yXd+
nJzpXSgV1vRBJW2CvFS1NqXrqzLFi4MXKwktkOGX0TpWc1G+2
A5uJ46T7zjqfTD5xFouC42r9elQJVUuLQOcBkFblrTquRQOEgP4qcVr5387NEFAGqlz0iEsZ807kOG9Anb0S
+hZ/
K4IeYRtobqTMFb6venCyMUgCRUQtPosWes8EEyaup8SP47zRnN4mV52ffdzoU6oT3KnZir+
q3QdZaT0QBu4KOOslUeLKHeQvoxfI/
RrIvEPCQDF9TJJrfGrqOgb90GV621h3a7o1lebK1WOKUoz6SMUuUdRcVkcFaYx+
h7DjDCq5u9HywfzxC6iRClZgR+F+
eht4MexuJi6eB5hNBADFe2nS5mKsAnXnPtYSKwHKj7eKDnW42sBOqXBac+f+IBR/0
yGODlhW9kl3dK+eEWfmZrLtSvlnUFhyDK5+aSSGWNTqqUljIOsoMRoW+qGYZZ2R1bJ40f/
M3G+
zJbzYD27bjaLQTv67fjpE8U4DWMXJsS6J3sAhsZXUSJREm4YpsWwNI8DPW7qTjiJyq4SaxOJBLZoxsK
/EehLifQIKGXL85PYCVMVr3jM+
BB7YfF2DOGg0sOnPQgV7qYyVcLsTeU230pqTTyrvpSEGpwOHiLzy+
bvLx9AhTR4Fy7MGDclkSEI2kpnIFxMlcQ6OKTt+
gDoDptKNowUNeJf3vpONQbWz1AJ9DhVFGmgKCuLo8It6isIH2LMilXzhJ2kkT4zzYX0ZtnG4Z6LiJL8X7U5n
/F2luQgm+rOoyG1WTajCy4dpDs7EKOX7l2GyUMOAcmSHlEHEFIzEWlv/

```
        timHdbvM3f0AKDKZrtL5crVdmGmEKMlkQPMsotZxXuZnazWTE6oarSlePhkNtJj3LO/
        LRhPYKwkplYewETpVhzwXg46FFV5sMrHw2wcl8pdsqcgME4SC2eFPT5hvkz+
        l12kaKwG2z0s68vNHaXKgd5OhzyUC9oigwYirJNHTLiDPKfUZPI1Xzu2Q8Xcz0aGP+
        zlIuGmM32bie5Y+RihmU0iBleKi5gKV8R39/
        crScJtCMrfB9L7HsJEtMKLwNgaDh5xqQO2F5z19erxCJYs0NOJ26qOIl0W94pIWuDVBqHEVDfvgBFxX
        /LHWrUGCEV7UeW8gUFjnZtfQJt48/7
        KeCTLQagpmFBCacCGSRUwo6Jaor4Bpy9LLX76jv8JC8k5g8vddAce8uzwSut3oPF2XVjRGVeJvDOK5t3C4kU
        +0k1UupedPGVX976Cpe7Aw7ugr3SKdiYF1HEgeHskenEutb80tdroPAn2YVP+XO+4md/
        YI7kwyC3BWlZS8ABek7Lma9fvMp8fcqkzcNnV/zJLly5F+RO/EX9g8CvRzskMgREbt+
        Pus4y/Lm+EY3dDeyvhxoK/fusnKWqvjEUatUJZrRpg+9
        bvTYy4CcCmXuBzpypyuaLkjcoIXfV3J0s5asX4/
        IsC9XHIX15Itq2d8JlayQYx1M7oV7mJ0C4Guk/
        rHfjvpo9hEAcVnzKr3MNCzUQmMTTHQH0ew08AdpMv0t48+xpzKDj8+SA/GeufTaEl56pp+/
        ZSKkiGW9fsRAaByy4kXRlF9FsdlB1wM9AnuYrTyB+mEQ5JuTkeEturQcs/w/
        vCZQFedZEVPllEP2qMTsQkIH2O8qmIxkBkjKKfdk2Sj0lk8xj4ozM7R3K6I1DbTn9cW2/
        qmKPtBPdm5N6EVU28tXThGuo309VEoJMl5cHqMntS/rvS1D0KON5DiWCOzfm7AsKs/
        NBYu7iM0/PRz50sCvRWHsEeFHIAuFNavneFmGqPTfhnu7W059e9j9JIqMw8l0zg+
        j0iQ7pSyBuGeeekptiNK74kHEushp94qzSczm7fT3bCbFOcq2cHIMNFwBjy1dV+
        Bnnnid1wuHqQreSvkN0z3Vdy5C9gqO0u/UL8cHRFZMYrcq6hiqpqS4lYHhI8ROqpYk/
        Oq99wu5jWpVp/NQ12vcxclx/H+Dbj4lYcAs6ycb/rko/zxs8/ohJtfrm4lDOPHETDybJhb/
        x1v35nJGCX7WjphpJy+PjxVbroC5T8x/B0ondgoLbcyXngAxl267KU13irRC/rTh+
        h1GFX2KzXwM1n3GrjcEntWR05HlG0IZ30z3p5QskvCzyssjo8XPdLtTZu5a5y67UR2pXvx1DG6uQ1jqMr1tI
        +32iHjPqbqqBrzCKbSZQHSBQftUeXQomsD3freeFI+3
        M4qZlSD8RHGVum9beEYfT0HgYCulLfS08fH6Ol30vSA18e8IN4GMMwqlcDT+8
        ftx2qXZncm6X3XDK1E2eYx1jc3b1fGQIARPGx+xBRI8U5L35e4q3my/
        AUScUaECMfP3k4PjAd2Kh/
        nK8YjWVON5eNA38jj0jjm0HDFLanxvYFKWQ7bUsL65pL99LoBKnSoqN61NM9wkzESzTTwC52lpcWkJyCks8k
        +2o2d43SjknLjZu94soFufW9bUYb+7+xLOpbAhPXuXtag4sP+BFz6KLgi4p2zZ/
        i7GO5RtGD3bKu4IP72jAcYQe7x24zZr+zcs4P0bbWHPCzrO3NfspZnGcJWfbC/
        LdeENaF1Ssd0hKXk78XWa/
        GSWwfhvlsgqfb82ebQybk34wHCKr1JSzBiXLPr6CnGVMVNIW0DCev/
        huqds5L2HmQ6rDLfEGH0dSjOoNEqLV0t949OGgSg8VwZDngjeUcPz2C0iu1zHl+
        kl8AjlOlspKGclnBL0hD2tor1f+
        ROOygbORJbV1bg7JRcwjo0b5Eeo5eL1aQNevoYvbGWPCUrwmNZa9cpHsnO/
        tbMqCKdVxSfydSKp2V36tYPQbhIIpxWck+i3Z
    Cqn90S+
        cqBvGxAODx1jD5vSVkHmouyagLxiIfIDXWku3QnzuVSLqfvipJw0bRVb9ZPQXgVrXVxKXuzrkzResUINaTv2
        +cNfpfFwUa59ZMUuqI5s7dDw3Ubt9NaMlvk9OPwGDDy9qHe4/
        PqiWN0vubnqkx0zPd1rXzQeJxM7+EhwkIS6q/27mw7bSAGA/
        AvjY2XGtcsCVCSspkQlqL3f7s22GaM7dg9p8tpg74bls0tmPFoJE17hBK0tpIL68v9J1dbv6kPhnAy0sIMO
        /
        tE2coXWXAWQN4a3J7Vv1gCDD5LZhEBcOO2BhS24N5ndCIaSCF4mRdJRHZHeyM5b06o0ChXHxDhNZBCNkzJyJ
        /c9hJZSpZnbvLf7wCA4T5Qmzv5EFex0ZoREivR+
        XdiBdiDU7rzaeoP9uHchWMoNRdeIs0I43MtjqhYWjT6GVMsWTy5tD1AYy1+
        jaTz3DlYlaUkRNtJbMHoxFjYY8Im63nKBDvyoMYa+99
        bcOu7gYBSehIjfOcAoSy4hE8qcYhjRyx2tPow8oMV3uPVnbGXqhtWdJbF+
        ONM7w5g3DkylzYj4MelLobxECyCIyUGG+3
        BhhWcxq9NMrIak2j80HeeLbXlIuT3Ag2QOdTugQTVBGQ+
        jft4QnpZyONvBdtHKHuCzPwuPoS+17gOE7gheNV1DCNoOh9HNcfwRlnsQ5tgT7Or9dw/
        TJqvQljHffC60g0FxWcOCIHFxYDq9iRKuNtI61RUXeH+RLI/YdZ8prO3xs6Mg/
        tPboK5lAK5gRuOYurBTqRu5NMd6doQhpIbp81Z6QfLq+
        bJxFTubxKDHxbeOYmyuPlXOctqLtEzGyjlgkNNkUDuQY0cSTnJN0resxUmXiec7rKvhlDx
        /54ElGRG5+cHDEiEkf1P7HRafy2XwkCz48H07VOT1L3jIh/IMI7JmJaWlV8s/fdQHgP7+
        VNUE61E2NZnmTYjvHNscfnu6+zY5quk+kg/0KCx+
        b9ymP68Ppw3AAAa5grVdKZRrcIZ5u1IjSzla5nu6YySou0mYG621p79ayZ/TBq3a+
        QRrlSrVb2pksD6oWlrrAdeXQ79ZgI7kksZ4LuSD8+
        l5NmcpM2e45a9it6602pDpz4fmBEEnBrD+
        dnMN7nDuQieEVu8kXKwghdGJ8OnjTyp3rIptQv6W3S2Wq/nIBae1aN4aLFWkxeajJxgV7/
        pRqyYyZ0IOBhHNQ27xIeJpodV+pPO4vJ0ugt6FxEp7+dLs/
```

```
            eNVi3oWSWYHQgBl6fYiMl3mDa1+N0pX7TsTzeF4UipqtdahSLkQojcqZ+IJkhGF0YwGi4Hz
            +HvueFnwdPp4cewLpxV+rPS40ECRitRmGxezfFGzFLAkptKBidmAC4j/3
            jet0ffYImx5X6a1bZhLd2o52IMeXlPE6yVlF5qWw4B6EbMQ0aNlPqr+NxEIHQoXfyr/
            VkIuItRwCVS1biCD+JLjRtptRfNTozuhDQfxkEcmEWX21BbG//lEn18FypfxhH6EYEzI+
            z1enrS7J+BEAocEZrxZX6AAhXmvlW6gMjIgD6eK2UUkoppZRS/7
            XvgFkg39rBJagAAAAASUVORK5CYII=" alt="Logo" width="162"
            height="172" /></p>
    <h2 style="text-align: center;"><span
            style="text-decoration: underline; color: #3598db; font-size: 24pt;">
                Conditions d'inscription</span></h2><br/><br/>
    <ol>
        <li style="list-style-type: none;">
            <ol>
                <li dir="ltr"><span style="font-size: 14pt;">Boris Gourdoux <?php
                    ?> <span
                            style="text-decoration: underline;"><strong>décline
                                toute
                                responsabilité</strong></span> en cas d'accident
                                    sur les personnes
                        présentes au moment des cours, propriétaires de chiens ou
                            autres personnes
                        présentes sur le site.<br /><br /></span></li>
                <li><span style="font-size: 14pt;">Les propriétaires des chiens
                    doivent être titulaires
                        d'un <span style="text-decoration: underline;"><strong>
                            contrat d'assurance
                                responsabilité civile vie privée</span></strong>,
                                    garantissant les
                        dommages corporels, matériels et immatériels causés aux
                            tiers par les
                        animaux domestiques, ainsi que le remboursement des frais v
                            étérinaires de
                        l'animal et du coût des certificats prescrits en cas de
                            morsure.<br /><br /></span></li>
                <li><span style="font-size: 14pt;">Boris Gourdoux <span style="text
                    -decoration: underline;"><strong>ne
                                pratique aucune sélection</span></strong> quant à
                                    la race ou à
                        l'âge des chiens pour l'inscription aux cours.<br /><br
                            /></span></li>
                <li><span style="font-size: 14pt;">Leurs enfants sont autorisés à
                    assister aux cours
                        <span style="text-decoration: underline;"><strong>sous la
                            surveillance et la
                                responsabilité de leurs parents</span></strong>.</
                                    span><br /><br /></li>
                <li><span style="font-size: 14pt;">Lors des cours collectifs vos
                    chiens <span
                                style="text-decoration: underline;"><strong>doivent
                                     être tenus en laisse</span></strong>
                        tant que l'éducateur ne vous a pas autorisé à les
                        lâcher.</span><br /><br /></li>
                <li><span style="font-size: 14pt;">Les chiens devront être <span
                                style="text-decoration: underline;"><strong>à jour
                                    de leurs vaccinations</span></strong>
                        avant le début des cours.</span><br /><br /></li>
                <li><span style="font-size: 14pt; text-decoration: underline;"><
                    strong>Les cours seront réglés sur place le jour
                         m&ecirc;me pour les cours à la carte. Pour les forfaits
```

```
, un acompte de 50% sera
                        versé lors de la réservation de ce dernier. Le solde, soit 50%, sera
                        versé au 3&egrave;me cours, sauf accord préalable avec
                        l'éducateur.</strong></span><br /><br /></li>
            <li><span style="font-size: 14pt;text-decoration: underline;"><strong>Tout abonnement commencé est dû et non
                        remboursable, quel que soit le motif de la rupture
                        anticipée.<br /><br /></strong></span></li>
            <li><span style="font-size: 14pt;"><span style="text-decoration: underline;"><strong>Boris Gourdoux ne
                            pourra en aucun cas être tenu responsable</strong></span> des mauvais
                    comportements et des non-acquis de mon chien en cas de non régularité du suivi des
                    séances de mon fait (au minimum une séance tout les 10 jours).</span><br /><br /><br /    >
            </li>
            <li><span style="font-size: 14pt;"><span style="text-decoration: underline;"><strong>Toute annulation de
                            cours individuels doit être formulée au moins 24h avant la
                            séance</strong></span>, faute de quoi, elle sera facturée et/ou
                    déduite du forfait.</span><br /><br /></li>
            <li><span style="font-size: 14pt;"><span style="text-decoration: underline;"><strong>TOUTE PUNITION
                            PHYSIQUE BASEE SUR LA PEUR ET/OU LA DOULEUR SUR LE CHIEN EST TOTALEMENT
                            PROHIBEE</strong></span>. <em>Le non-respect de l'animal et de cette clause mettra
                        fin immédiatement au contrat conclu entre Boris Gourdoux et le propriétaire du
                        chien, sans remboursement possible</em>.</span><br /><br /></li>
            <li><span style="font-size: 14pt;"><span style="text-decoration: underline;"><strong>L'UTILISATION
                            DE COLLIER ETRANGLEUR, A POINTE OU NON, COLLIER ELECTRIQUE ET/OU TOUT OBJET POUVANT
                            INFLIGER UNE DOULEUR AU CHIEN EST RIGOUREUSEMENT INTERDITE</strong></span>. <em>Le
                        non-respect de l'animal et de cette clause mettra fin immédiatement au contrat
                        conclu entre Boris Gourdoux et le propriétaire du chien, sans remboursement
                        possible</em>.</span></li>
        </ol>
    </li>
</ol>
<p style="padding-left: 40px;"><span style="font-size: 14pt;">Choix du forfait :</span></p>
<ul>
    <li style="list-style-type: none;">
        <ul>
        <?php
            switch ($package_number) {
                case 1:
                    echo '<li><span style="font-size: 14pt;">Bilan d\'évaluation : 70 &euro; / 80 CHF</span></li>';
```

```
                        break;
                    case 2:
                        echo '<li><span style="font-size: 14pt;">Bilan + 1 séance d
                           \'éducation : 125 &euro; / 140 CHF</span></li>';
                        break;
                    case 3:
                        echo ' <li><span style="font-size: 14pt;">Forfait bilan + 3
                            séances : 230 &euro; / 250 CHF</span></li>';
                        break;
                    case 4:
                        echo ' <li><span style="font-size: 14pt;">Forfait bilan + 6
                            séances : 400 &euro; / 440 CHF</span></li>';
                        break;
                    case 5:
                        echo '<li><span style="font-size: 14pt;">Forfait bilan + 9
                           séances : 520 &euro; / 560 CHF</span></li>';
                        break;

                    default:
                        echo "";
                        break;
                }
            ?>
            </ul>
        </li>
    </ul>
    <p style="padding-left: 40px;"><span style="font-size: 14pt;">Pour faire valoir
         ce que de
            droit.</span><br /><br /><span style="font-size: 14pt;">Date et
                signature précédée de
            la mention &laquo; lu et approuvé &laquo;</span></p>
    <p>&nbsp;</p>
    <h3 style="padding-left: 120px;">Lu et approuvé par <?= "$userfirstname
        $userlastname" ?></h3>
    <p style="padding-left: 120px;">
        <figure>
            <img src="<?= $signature_base64 ?>" width="150" height="150">
            <figcaption><?= $date ?></figcaption>
        </figure>
    </p>
</body>

</html>
```

Listing 50 – ./Sources/resources/template/iCalendar//textunderscoreappointment.php

```
<?php
/**
 * iCalendar_appointment.php.php
 *
 * Template of the content of an ics file (iCalendar) used in the HelperController
     with the sendMailWithICSFile function
 *
 * @author  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
 */
header('Content-type: text/calendar; charset=utf-8');
header('Content-Disposition: attachment; filename=' . $filename);

$now = new DateTime('NOW');
```

```

echo "BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//douceurdechien/handcal//NONSGML v1.0//FR
CALSCALE:GREGORIAN
METHOD:PUBLISH
BEGIN:VEVENT
UID:" . md5(time()) . "
DTSTAMP;TZID=/Europe/Berlin:" . gmdate("Ymd\THis",$now->getTimestamp() + 60*60*2) .
    "
DTSTART;TZID=/Europe/Berlin:".gmdate("Ymd\THis",$start_datetime->getTimestamp() +
    60*60*2)."
DTEND;TZID=/Europe/Berlin:".gmdate("Ymd\THis",$end_datetime->getTimestamp() +
    60*60*2)."
SUMMARY:Rendez-vous Douceur de Chien
LOCATION:701 Avenue de la Bigorre, 31210 Montréjeau, France
ORGANIZER:MAILTO:douceurdechien@douceurdechien.com
END:VEVENT
END:VCALENDAR";
```