

# 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

```

1  <?php
2  /**
3   * AbsenceController.php
4   *
5   * Controller of the Absence model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAOAbsence;
13 use App\DataAccessObject\DAOUser;
14 use App\Controllers\ResponseController;
15 use App\Controllers\HelperController;
16 use App\Models\Absence;
17 use App\System\Constants;
18
19 class AbsenceController {
20
21     private DAOAbsence $DAOAbsence;
22     private DAOUser $DAOUser;
23
24     /**
25      *
26      * Constructor of the AbsenceController object.
27      *
28      * @param PDO $db The database connection
29      */
30     public function __construct(\PDO $db)
31     {
32         $this->DAOAbsence = new DAOAbsence($db);
33         $this->DAOUser = new DAOUser($db);
34     }
35
36     /**
37      *
38      * Method to return all absences in JSON format.
39      *
40      * @return string The status and the body in json format of the response
41      */
42     public function getAllAbsences()
43     {
44         $headers = apache_request_headers();
45
46         if (!isset($headers['Authorization'])) {
47             return ResponseController::notFoundAuthorizationHeader();
48         }
49
50         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
51
52         if (is_null($userAuth) || $userAuth->code_role != Constants::
53             ADMIN_CODE_ROLE) {
54             return ResponseController::unauthorizedUser();
55         }
56
57         $allAbsences = $this->DAOAbsence->findAll(false,$userAuth->id);

```

```

58         return ResponseController::successfulRequest($allAbsences);
59     }
60
61     /**
62      *
63      * Method to return a absence in JSON format.
64      *
65      * @param int $id The absence identifier
66      * @return string The status and the body in JSON format of the response
67      */
68     public function getAbsence(int $id)
69     {
70         $headers = apache_request_headers();
71
72         if (!isset($headers['Authorization'])) {
73             return ResponseController::notFoundAuthorizationHeader();
74         }
75
76         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
77
78         if (is_null($userAuth) || $userAuth->code_role != Constants::
79             ADMIN_CODE_ROLE) {
80             return ResponseController::unauthorizedUser();
81         }
82
83         $absence = $this->DAOAbsence->find($id,$userAuth->id);
84
85         if (is_null($absence)) {
86             return ResponseController::notFoundResponse();
87         }
88
89         return ResponseController::successfulRequest($absence);
90     }
91
92     /**
93      *
94      * Method to create a absence.
95      *
96      * @param Absence $absence The absence model object
97      * @return string The status and the body in JSON format of the response
98      */
99     public function createAbsence(Absence $absence)
100     {
101         $headers = apache_request_headers();
102
103         if (!isset($headers['Authorization'])) {
104             return ResponseController::notFoundAuthorizationHeader();
105         }
106
107         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
108
109         if (is_null($userAuth) || $userAuth->code_role != Constants::
110             ADMIN_CODE_ROLE) {
111             return ResponseController::unauthorizedUser();
112         }
113
114         $absence->id_educator = $userAuth->id;
115
116         if (!$this->validateAbsence($absence)) {

```

```

115         return ResponseController::unprocessableEntityResponse();
116     }
117
118     if (!HelperController::validateDateFormat($absence->date_absence_from) || !
119         HelperController::validateDateFormat($absence->date_absence_to) ) {
120         return ResponseController::invalidDateFormat();
121     }
122
123     if (!HelperController::validateChornologicalTime($absence->
124         date_absence_from,$absence->date_absence_to)) {
125         return ResponseController::chronologicalDateProblem();
126     }
127
128     $this->DAOAbsence->insert($absence);
129
130     return ResponseController::successfulCreatedRessource();
131 }
132
133 /**
134  *
135  * Method to update a absence.
136  *
137  * @param Absence $absence The absence model object
138  * @return string The status and the body in JSON format of the response
139  */
140 public function updateAbsence(Absence $absence)
141 {
142     $headers = apache_request_headers();
143
144     if (!isset($headers['Authorization'])) {
145         return ResponseController::notFoundAuthorizationHeader();
146     }
147
148     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
149
150     if (is_null($userAuth) || $userAuth->code_role != Constants::
151         ADMIN_CODE_ROLE) {
152         return ResponseController::unauthorizedUser();
153     }
154
155     $actualAbsence = $this->DAOAbsence->find($absence->id,$userAuth->id);
156
157     if (is_null($actualAbsence)) {
158         return ResponseController::notFoundResponse();
159     }
160
161     $actualAbsence->date_absence_from = $absence->date_absence_from ??
162         $actualAbsence->date_absence_from;
163     $actualAbsence->date_absence_to = $absence->date_absence_to ??
164         $actualAbsence->date_absence_to;
165     $actualAbsence->description = $absence->description ?? $actualAbsence->
166         description;
167
168     if (!HelperController::validateDateFormat($actualAbsence->date_absence_from
169 ) || !HelperController::validateDateFormat($actualAbsence->
170     date_absence_to) ) {
171         return ResponseController::invalidDateFormat();
172     }

```

```

166         if (!HelperController::validateChornologicalTime($actualAbsence->
167             date_absence_from,$actualAbsence->date_absence_to)) {
168             return ResponseController::chronologicalDateProblem();
169         }
170         $this->DAOAbsence->update($actualAbsence);
171
172         return ResponseController::successfulRequest(null);
173     }
174
175     /**
176      *
177      * Method to delete a absence.
178      *
179      * @param int $id The absence identifier
180      * @return string The status and the body in JSON format of the response
181      */
182     public function deleteAbsence($id)
183     {
184         $headers = apache_request_headers();
185
186         if (!isset($headers['Authorization'])) {
187             return ResponseController::notFoundAuthorizationHeader();
188         }
189
190         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
191
192         if (is_null($userAuth) || $userAuth->code_role != Constants::
193             ADMIN_CODE_ROLE) {
194             return ResponseController::unauthorizedUser();
195         }
196
197         $absence = $this->DAOAbsence->find($id,$userAuth->id);
198
199         if (is_null($absence)) {
200             return ResponseController::notFoundResponse();
201         }
202
203         $this->DAOAbsence->delete($absence);
204
205         return ResponseController::successfulRequest(null);
206     }
207
208     /**
209      *
210      * Method to check if the absence required fields have been defined for the
211      * creation.
212      *
213      * @param Absence $absence The absence model object
214      * @return bool
215      */
216     private function validateAbsence(Absence $absence)
217     {
218         if ($absence->date_absence_from == null) {
219             return false;
220         }
221
222         if ($absence->date_absence_to == null) {
223             return false;

```

```

222     }
223
224     if ($absence->id_educator == null) {
225         return false;
226     }
227
228     return true;
229 }
230 }

```

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

```

1  <?php
2  /**
3   * AppointmentController.php
4   *
5   * Controller of the Appointment model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAOAppointment;
13 use App\DataAccessObject\DAOTimeSlot;
14 use App\DataAccessObject\DAOUser;
15 use App\Controllers\ResponseController;
16 use App\Controllers\HelperController;
17 use App\Models\Appointment;
18 use App\System\Constants;
19
20 class AppointmentController {
21
22     private DAOAppointment $DAOAppointment;
23     private DAOTimeSlot $DAOTimeSlot;
24     private DAOUser $DAOUser;
25
26
27     /**
28      *
29      * Constructor of the AppointmentController object.
30      *
31      * @param PDO $db The database connection
32      */
33     public function __construct(\PDO $db)
34     {
35         $this->DAOAppointment = new DAOAppointment($db);
36         $this->DAOTimeSlot = new DAOTimeSlot($db);
37         $this->DAOUser = new DAOUser($db);
38     }
39
40     /**
41      *
42      * Method to return all Appointments in JSON format.
43      *
44      * @return string The status and the body in json format of the response
45      */
46     public function getAllAppointments()
47     {

```

```

48     $headers = apache_request_headers();
49
50     if (!isset($headers['Authorization'])) {
51         return ResponseController::notFoundAuthorizationHeader();
52     }
53
54     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
55
56     if (is_null($userAuth)) {
57         return ResponseController::unauthorizedUser();
58     }
59
60     if ($userAuth->code_role == Constants::ADMIN_CODE_ROLE){
61         $allAppointments = $this->DAOAppointment->findAll(null,$userAuth->id);
62     }
63     else{
64         $allAppointments = $this->DAOAppointment->findByUserId($userAuth->id);
65     }
66
67     return ResponseController::successfulRequest($allAppointments);
68 }
69
70 /**
71  *
72  * Method to return a appointment in JSON format.
73  *
74  * @param int $id The appointment identifier
75  * @return string The status and the body in JSON format of the response
76  */
77 public function getAppointment(int $id)
78 {
79     $headers = apache_request_headers();
80
81     if (!isset($headers['Authorization'])) {
82         return ResponseController::notFoundAuthorizationHeader();
83     }
84
85     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
86
87     if (is_null($userAuth) || $userAuth->code_role != Constants::
88         ADMIN_CODE_ROLE) {
89         return ResponseController::unauthorizedUser();
90     }
91
92     $appointment = $this->DAOAppointment->find($id);
93
94     if (is_null($appointment)) {
95         return ResponseController::notFoundResponse();
96     }
97
98     return ResponseController::successfulRequest($appointment);
99 }
100
101 /**
102  *
103  * Method to create a appointment.
104  *
105  * @param Appointment $appointment The appointment model object
106  * @return string The status and the body in JSON format of the response

```



```

106     */
107     public function createAppointment(Appointment $appointment)
108     {
109         $headers = apache_request_headers();
110
111         if (!isset($headers['Authorization'])) {
112             return ResponseController::notFoundAuthorizationHeader();
113         }
114
115         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
116
117         if (is_null($userAuth)) {
118             return ResponseController::unauthorizedUser();
119         }
120
121         if (!$this->validateAppointment($appointment)) {
122             return ResponseController::unprocessableEntityResponse();
123         }
124
125         $userCustomer = $this->DAOUser->find($appointment->user_id_customer);
126         $userEducator = $this->DAOUser->find($appointment->user_id_educator);
127
128         if (is_null($userCustomer) || is_null($userEducator) || $userCustomer->
            code_role != Constants::USER_CODE_ROLE || $userEducator->code_role !=
            Constants::ADMIN_CODE_ROLE) {
129             return ResponseController::notFoundResponse();
130         }
131
132         if (!HelperController::validateDateTimeFormat($appointment->
            datetime_appointment)) {
133             return ResponseController::invalidDateTimeFormat();
134         }
135
136         $datetime_start = new \DateTime($appointment->datetime_appointment, new \
            DateTimeZone("Europe/Berlin"));
137         $datetime_end = clone $datetime_start;
138         $datetime_end->modify('+'. $appointment->duration_in_hour. " hours");
139
140         $date = $datetime_start->format("Y-m-d");
141         $time_start = $datetime_start->format("H:i:s");
142         $time_end = $datetime_end->format("H:i:s");
143
144         if ($userAuth->code_role == Constants::USER_CODE_ROLE) {
145
146             if (!$this->DAOTimeSlot->findAppointmentSlotsForEducator($date,
                $time_start, $time_end, $appointment->user_id_educator)) {
147                 return ResponseController::invalidAppointment();
148             }
149
150         }
151
152         $this->DAOAppointment->insert($appointment);
153
154         $filename = "iCal-" . $datetime_start->format("Ymd") . ".ics";
155
156         $educator_fullname = $userEducator->firstname . " " . $userEducator->
            lastname;
157
158         HelperController::sendMailWithICSFile($datetime_start, $datetime_end,

```

```

        $educator_fullname, $userCustomer->email, $filename);
159
160     return ResponseController::successfulCreatedResource();
161 }
162
163 /**
164  *
165  * Method to update a appointment.
166  *
167  * @param int $id The appointment identifier
168  * @return string The status and the body in JSON format of the response
169  */
170 public function updateAppointment(Appointment $appointment)
171 {
172     $headers = apache_request_headers();
173
174     if (!isset($headers['Authorization'])) {
175         return ResponseController::notFoundAuthorizationHeader();
176     }
177
178     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
179
180     if (is_null($userAuth) || $userAuth->code_role != Constants::
181         ADMIN_CODE_ROLE) {
182         return ResponseController::unauthorizedUser();
183     }
184
185     $actualAppointment = $this->DAOAppointment->find($appointment->id);
186
187     if (is_null($actualAppointment)) {
188         return ResponseController::notFoundResponse();
189     }
190
191     $actualAppointment->note_text = $appointment->note_text ??
192         $actualAppointment->note_text;
193     $actualAppointment->summary = $appointment->summary ?? $actualAppointment->
194         summary;
195
196     $this->DAOAppointment->update($actualAppointment);
197
198     return ResponseController::successfulRequest(null);
199 }
200
201 /**
202  *
203  * Method to delete a appointment.
204  *
205  * @param int $id The appointment identifier
206  * @return string The status and the body in JSON format of the response
207  */
208 public function deleteAppointment(int $id)
209 {
210     $headers = apache_request_headers();
211
212     if (!isset($headers['Authorization'])) {
213         return ResponseController::notFoundAuthorizationHeader();
214     }
215
216     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

```

```

214
215     if (is_null($userAuth)) {
216         return ResponseController::unauthorizedUser();
217     }
218
219     $appointment = $this->DAOAppointment->find($id);
220
221     if (is_null($appointment)) {
222         return ResponseController::notFoundResponse();
223     }
224
225     if ($userAuth->id == $appointment->user_id_educator xor $userAuth->id !=
        $appointment->user_id_customer ) {
226         return ResponseController::unauthorizedUser();
227     }
228
229     $this->DAOAppointment->delete($appointment->id,$userAuth->id);
230
231     return ResponseController::successfulRequest(null);
232 }
233
234 /**
235  *
236  * Method to upload a graphical note.
237  *
238  * @return string The status and the body in JSON format of the response
239  */
240 public function uploadNoteGraphical()
241 {
242     $headers = apache_request_headers();
243
244     if (!isset($headers['Authorization'])) {
245         return ResponseController::notFoundAuthorizationHeader();
246     }
247
248     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
249
250     if (is_null($userAuth) || $userAuth->code_role != Constants::
        ADMIN_CODE_ROLE) {
251         return ResponseController::unauthorizedUser();
252     }
253
254     if (!isset($_FILES["note_graphical"]) || !is_uploaded_file($_FILES["
        note_graphical"]["tmp_name"]) || !isset($_POST["appointment_id"])) {
255         return ResponseController::unprocessableEntityResponse();
256     }
257
258     $appointment = $this->DAOAppointment->find($_POST["appointment_id"]);
259
260     if (is_null($appointment)) {
261         return ResponseController::notFoundResponse();
262     }
263
264     switch ($_FILES["note_graphical"]["type"]) {
265         case Constants::IMAGE_TYPE_PNG:
266             HelperController::pngTojpegConverter($_FILES["note_graphical"]["
                tmp_name"]);
267             break;
268         case Constants::IMAGE_TYPE_JPEG:

```

```

269         break;
270     default:
271         return ResponseController::imageFileFormatProblem();
272         break;
273     }
274
275     $tmp_file = $_FILES["note_graphical"]["tmp_name"];
276     $img_name = HelperController::generateRandomString();
277     $upload_dir = HelperController::getDefaultDirectory()."storage/app/
graphical_note/".$img_name.".jpeg";
278
279     if (!is_null($appointment->note_graphical_serial_id)) {
280         $filename = HelperController::getDefaultDirectory()."storage/app/
graphical_note/".$appointment->note_graphical_serial_id.".jpeg";
281         if (file_exists($filename)) {
282             unlink($filename);
283         }
284     }
285
286     if (!move_uploaded_file($tmp_file,$upload_dir)) {
287         return ResponseController::uploadFailed();
288     }
289
290     $appointment->note_graphical_serial_id = $img_name;
291
292     $this->DAOAppointment->update($appointment);
293
294     return ResponseController::successfulRequest();
295 }
296
297 /**
298  *
299  * Method to download a graphical note.
300  *
301  * @param string $serial_id The serial_id of the graphical note
302  * @return string The status and the body in JSON format of the response
303  */
304 public function downloadNoteGraphical(string $serial_id)
305 {
306     $headers = apache_request_headers();
307
308     if (!isset($headers['Authorization'])) {
309         return ResponseController::notFoundAuthorizationHeader();
310     }
311
312     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
313
314     if (is_null($userAuth) || $userAuth->code_role != Constants::
ADMIN_CODE_ROLE) {
315         return ResponseController::unauthorizedUser();
316     }
317
318     if (is_null($this->DAOAppointment->findBySerialId($serial_id))) {
319         return ResponseController::notFoundResponse();
320     }
321
322     $image = HelperController::getDefaultDirectory()."storage/app/
graphical_note/".$serial_id.".jpeg";
323

```

```

324         $imageInfo = getimagesize($image);
325
326         header("Content-Type: " . $imageInfo["mime"]);
327         header("Content-Length: " . filesize($image));
328         readfile($image);
329
330         return ResponseController::successfulRequest();
331     }
332
333     /**
334     *
335     * Method to check if the appointment required fields have been defined for the
336     * creation.
337     *
338     * @param Appointment $appointment The appointment model object
339     * @return bool
340     */
341     private function validateAppointment(Appointment $appointment)
342     {
343         if ($appointment->datetime_appointment == null) {
344             return false;
345         }
346
347         if ($appointment->duration_in_hour == null) {
348             return false;
349         }
350
351         if ($appointment->user_id_customer == null) {
352             return false;
353         }
354
355         if ($appointment->user_id_educator == null) {
356             return false;
357         }
358
359         return true;
360     }
361 }

```

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

```

1 <?php
2 /**
3  * DocumentController.php
4  *
5  * Controller of the Document model.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAODocument;
13 use App\DataAccessObject\DAOUser;
14 use App\Controllers\ResponseController;
15 use App\Controllers\HelperController;
16 use App\Models\Document;
17 use App\System\Constants;
18

```

```

19 class DocumentController {
20
21     private DAODocument $DAODocument;
22     private DAOUser $DAOUser;
23
24     /**
25      *
26      * Constructor of the DocumentController object.
27      *
28      * @param PDO $db The database connection
29      */
30     public function __construct(\PDO $db)
31     {
32         $this->DAODocument = new DAODocument($db);
33         $this->DAOUser = new DAOUser($db);
34     }
35
36     /**
37      *
38      * Method to return all documents in JSON format.
39      *
40      * @return string The status and the body in json format of the response
41      */
42     public function getAllDocuments()
43     {
44         $headers = apache_request_headers();
45
46         if (!isset($headers['Authorization'])) {
47             return ResponseController::notFoundAuthorizationHeader();
48         }
49
50         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
51
52         if (is_null($userAuth) || $userAuth->code_role != Constants::
53             ADMIN_CODE_ROLE) {
54             return ResponseController::unauthorizedUser();
55         }
56
57         $allDocuments = $this->DAODocument->findAll();
58
59         return ResponseController::successfulRequest($allDocuments);
60     }
61
62     /**
63      *
64      * Method to return a document in JSON format.
65      *
66      * @param int $id The document identifier
67      * @return string The status and the body in JSON format of the response
68      */
69     public function getDocument(int $id)
70     {
71         $headers = apache_request_headers();
72
73         if (!isset($headers['Authorization'])) {
74             return ResponseController::notFoundAuthorizationHeader();
75         }
76
77         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

```

```

77
78         if (is_null($userAuth) || $userAuth->code_role != Constants::
           ADMIN_CODE_ROLE) {
79             return ResponseController::unauthorizedUser();
80         }
81
82         $document = $this->DAODocument->find($id);
83
84         if (is_null($document)) {
85             return ResponseController::notFoundResponse();
86         }
87
88         return ResponseController::successfulRequest($document);
89     }
90
91     /**
92      *
93      * Method to create a document.
94      *
95      * @param Document $document The document model object
96      * @return string The status and the body in JSON format of the response
97      */
98     public function createDocument(Document $document)
99     {
100         $headers = apache_request_headers();
101
102         if (!isset($headers['Authorization'])) {
103             return ResponseController::notFoundAuthorizationHeader();
104         }
105
106         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
107
108         if (is_null($userAuth) || $userAuth->code_role != Constants::
           ADMIN_CODE_ROLE) {
109             return ResponseController::unauthorizedUser();
110         }
111
112         if (!$this->validateDocument($document)) {
113             return ResponseController::unprocessableEntityResponse();
114         }
115
116         if (!HelperController::validateDocumentTypeFormat($document->type)) {
117             return ResponseController::invalidDocumentTypeFormat();
118         }
119
120         $user = $this->DAOUser->find($document->user_id);
121
122         if (is_null($user)) {
123             return ResponseController::notFoundResponse();
124         }
125
126         $filename = HelperController::generateRandomString();
127
128         switch ($document->type) {
129             case Constants::DOCUMENT_TYPE_CONDITIONS_OF_REGISTRATION:
130
131                 if (!$this->validateDocumentConditionsRegistration($document)) {
132                     return ResponseController::unprocessableEntityResponse();
133                 }

```

```

134
135         if (!HelperController::validatePackageNumber($document->
136             package_number)) {
137             return ResponseController::packageNumberFormatProblem();
138         }
139
140         $package_number = $document->package_number;
141         $date = date('d/m/Y');
142         $document->document_serial_id = $filename;
143         $signature_base64 = $document->signature_base64;
144         $userfirstname = $user->firstname;
145         $userlastname = $user->lastname;
146
147         HelperController::storeConditionsRegistration($filename,
148             $package_number, $date, $signature_base64, $userfirstname,
149             $userlastname);
150
151         $upload_file = HelperController::getDefaultDirectory()."storage/app
152             /conditions_registration/".$filename.".pdf";
153
154         break;
155
156     case Constants::DOCUMENT_TYPE_POSTER:
157
158         if (!isset($_FILES["document"])) {
159             return ResponseController::unprocessableEntityResponse();
160         }
161
162         if ($_FILES["document"]["type"] != Constants::TYPE_DOCUMENT_PDF) {
163             return ResponseController::documentTypeNotPdfProblem();
164         }
165
166         $tmp_file = $_FILES["document"]["tmp_name"];
167         $upload_file = HelperController::getDefaultDirectory()."storage/app
168             /pdf/".$filename.".pdf";
169
170         if (!move_uploaded_file($tmp_file, $upload_file)) {
171             return ResponseController::uploadFailed();
172         }
173
174         break;
175     default:
176         return ResponseController::invalidDocumentTypeFormat();
177         break;
178 }
179
180 $document->document_serial_id = $filename;
181
182 $this->DAODocument->insert($document);
183
184 HelperController::sendMail("Bonjour et merci de faire confiance à la société
185     é Douceur de Chien, vous trouverez ci-joint le document qui a été ajout
186     é à votre compte, vous pouvez également accéder à ce document depuis
187     votre compte.", "Un nouveau document a été ajouté à votre compte", $user
188     ->email, null, $upload_file);
189
190 return ResponseController::successfulCreatedRessource();
191 }
192
193

```



```

184  /**
185  *
186  * Method to update a document.
187  *
188  * @param Document $document The document model object
189  * @return string The status and the body in JSON format of the response
190  */
191  public function updateDocument(Document $document)
192  {
193      $headers = apache_request_headers();
194
195      if (!isset($headers['Authorization'])) {
196          return ResponseController::notFoundAuthorizationHeader();
197      }
198
199      $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
200
201      if (!$userAuth || $userAuth->code_role != Constants::ADMIN_CODE_ROLE) {
202          return ResponseController::unauthorizedUser();
203      }
204
205      $actualDocument = $this->DAODocument->find($document->id);
206
207      if (is_null($actualDocument)) {
208          return ResponseController::notFoundResponse();
209      }
210
211      $actualDocument->document_serial_id = $document->document_serial_id ??
212          $actualDocument->document_serial_id;
213      $actualDocument->type = $document->type ?? $actualDocument->type;
214
215      if (!HelperController::validateDocumentTypeFormat($actualDocument->type)) {
216          return ResponseController::invalidDocumentTypeFormat();
217      }
218
219      $this->DAODocument->update($actualDocument);
220
221      return ResponseController::successfulRequest(null);
222  }
223  /**
224  *
225  * Method to delete a document.
226  *
227  * @param int $id The document identifier
228  * @return string The status and the body in JSON format of the response
229  */
230  public function deleteDocument(int $id)
231  {
232      $headers = apache_request_headers();
233
234      if (!isset($headers['Authorization'])) {
235          return ResponseController::notFoundAuthorizationHeader();
236      }
237
238      $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
239
240      if (!$userAuth || $userAuth->code_role != Constants::ADMIN_CODE_ROLE) {
241          return ResponseController::unauthorizedUser();

```

```

242     }
243
244     $document = $this->DAODocument->find($id);
245
246     if (is_null($document)) {
247         return ResponseController::notFoundResponse();
248     }
249
250     switch ($document->type) {
251         case Constants::DOCUMENT_TYPE_CONDITIONS_OF_REGISTRATION:
252             $filename = HelperController::getDefaultDirectory()."storage/app/
conditions_registration/".$document->document_serial_id.".pdf";
253
254             if (file_exists($filename)) {
255                 unlink($filename);
256             }
257             break;
258
259         case Constants::DOCUMENT_TYPE_POSTER:
260
261             $filename = HelperController::getDefaultDirectory()."storage/app/
pdf/".$document->document_serial_id.".pdf";
262
263             if (file_exists($filename)) {
264                 unlink($filename);
265             }
266             break;
267
268         default:
269             break;
270     }
271
272     $this->DAODocument->delete($document);
273
274     return ResponseController::successfulRequest(null);
275 }
276
277 /**
278  *
279  * Method to download a document.
280  *
281  * @param int $serial_id The document identifier
282  * @return string The status and the body in JSON format of the response
283  */
284 public function downloadDocument(string $serial_id)
285 {
286     $headers = apache_request_headers();
287
288     if (!isset($headers['Authorization'])) {
289         return ResponseController::notFoundAuthorizationHeader();
290     }
291
292     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
293
294     if (is_null($userAuth)) {
295         return ResponseController::notFoundResponse();
296     }
297
298     if ($userAuth || $userAuth->code_role == Constants::ADMIN_CODE_ROLE) {

```

```

299         $document = $this->DAODocument->findBySerialId($serial_id);
300     }
301     else{
302         $document = $this->DAODocument->findByUserIdAndSerialId($userAuth->id,
303             $serial_id);
304     }
305     if (is_null($document)) {
306         return ResponseController::notFoundResponse();
307     }
308
309     header("Content-Type: application/pdf");
310
311     switch ($document->type) {
312         case Constants::DOCUMENT_TYPE_CONDITIONS_OF_REGISTRATION:
313
314             $document_data = file_get_contents(HelperController::
315                 getDefaultDirectory()."storage/app/conditions_registration/".$
316                 $serial_id.".pdf");
317             break;
318
319         case Constants::DOCUMENT_TYPE_POSTER:
320
321             $document_data = file_get_contents(HelperController::
322                 getDefaultDirectory()."storage/app/pdf/".$serial_id.".pdf");
323             break;
324
325         default:
326             break;
327     }
328
329     return ResponseController::successfulRequestWithoutJson($document_data);
330 }
331
332 /**
333  *
334  * Method to check if the document required fields have been defined for the
335  * creation.
336  *
337  * @param Document $document The document model object
338  * @return bool
339  */
340 private function validateDocument(Document $document)
341 {
342     if ($document->type == null) {
343         return false;
344     }
345
346     if ($document->user_id == null) {
347         return false;
348     }
349
350     return true;
351 }
352
353 /**
354  *
355  * Method to check if the conditions registration document required fields have
356  * been defined for the creation.

```

```

352     *
353     * @param Document $document The document model object
354     * @return bool
355     */
356     private function validateDocumentConditionsRegistration(Document $document)
357     {
358         if ($document->package_number == null) {
359             return false;
360         }
361
362         if ($document->signature_base64 == null) {
363             return false;
364         }
365
366         return true;
367     }
368 }

```

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

```

1  <?php
2  /**
3   * DogController.php
4   *
5   * Controller of the Dog model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAODog;
13 use App\DataAccessObject\DAOUser;
14 use App\Controllers\ResponseController;
15 use App\Controllers\HelperController;
16 use App\Models\Dog;
17 use App\System\Constants;
18
19 class DogController {
20
21     private DAODog $DAODog;
22     private DAOUser $DAOUser;
23
24
25     /**
26     *
27     * Constructor of the DogController object.
28     *
29     * @param PDO $db The database connection
30     */
31     public function __construct(\PDO $db)
32     {
33         $this->DAODog = new DAODog($db);
34         $this->DAOUser = new DAOUser($db);
35     }
36
37     /**
38     *
39     * Method to return all dogs in JSON format.

```

```

40  *
41  * @return string The status and the body in json format of the response
42  */
43  public function getAllDogs()
44  {
45      $headers = apache_request_headers();
46
47      if (!isset($headers['Authorization'])) {
48          return ResponseController::notFoundAuthorizationHeader();
49      }
50
51      $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
52
53      if (is_null($userAuth) || $userAuth->code_role != Constants::
54          ADMIN_CODE_ROLE) {
55          return ResponseController::unauthorizedUser();
56      }
57
58      $allDogs = $this->DAODog->findAll();
59
60      return ResponseController::successfulRequest($allDogs);
61  }
62
63  /**
64   *
65   * Method to return a dog in JSON format.
66   *
67   * @param int $id The dog identifier
68   * @return string The status and the body in JSON format of the response
69   */
70  public function getDog(int $id)
71  {
72      $headers = apache_request_headers();
73
74      if (!isset($headers['Authorization'])) {
75          return ResponseController::notFoundAuthorizationHeader();
76      }
77
78      $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
79
80      if (is_null($userAuth) || $userAuth->code_role != Constants::
81          ADMIN_CODE_ROLE) {
82          return ResponseController::unauthorizedUser();
83      }
84
85      $dog = $this->DAODog->find($id);
86
87      if (is_null($dog)) {
88          return ResponseController::notFoundResponse();
89      }
90
91      return ResponseController::successfulRequest($dog);
92  }
93
94  /**
95   *
96   * Method to create a dog.
97   *
98   * @param Dog $dog The dog model object

```

```

97      * @return string The status and the body in JSON format of the response
98      */
99      public function createDog(Dog $dog)
100     {
101         $headers = apache_request_headers();
102
103         if (!isset($headers['Authorization'])) {
104             return ResponseController::notFoundAuthorizationHeader();
105         }
106
107         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
108
109         if (is_null($userAuth) || $userAuth->code_role != Constants::
110             ADMIN_CODE_ROLE) {
111             return ResponseController::unauthorizedUser();
112         }
113
114         if (!$this->validateDog($dog)) {
115             return ResponseController::unprocessableEntityResponse();
116         }
117
118         $user = $this->DAOUser->find($dog->user_id);
119
120         if (is_null($user)) {
121             return ResponseController::notFoundResponse();
122         }
123
124         $this->DAODog->insert($dog);
125
126         return ResponseController::successfulCreatedRessource();
127     }
128
129     /**
130      *
131      * Method to update a dog.
132      *
133      * @param Dog $dog The dog model object
134      * @return string The status and the body in JSON format of the response
135      */
136     public function updateDog(Dog $dog)
137     {
138         $headers = apache_request_headers();
139
140         if (!isset($headers['Authorization'])) {
141             return ResponseController::notFoundAuthorizationHeader();
142         }
143
144         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
145
146         if (is_null($userAuth) || $userAuth->code_role != Constants::
147             ADMIN_CODE_ROLE) {
148             return ResponseController::unauthorizedUser();
149         }
150
151         $actualDog = $this->DAODog->find($dog->id);
152
153         if (is_null($actualDog)) {
154             return ResponseController::notFoundResponse();
155         }

```

```

154
155     $actualDog->name = $dog->name ?? $actualDog->name;
156     $actualDog->breed = $dog->breed ?? $actualDog->breed;
157     $actualDog->sex = $dog->sex ?? $actualDog->sex;
158     $actualDog->picture_serial_id = $dog->picture_serial_id ?? $actualDog->
        picture_serial_id;
159     $actualDog->chip_id = $dog->chip_id ?? $actualDog->chip_id;
160
161     $this->DAODog->update($actualDog);
162
163     return ResponseController::successfulRequest(null);
164 }
165
166 /**
167  *
168  * Method to delete a dog.
169  *
170  * @param int $id The dog identifier
171  * @return string The status and the body in JSON format of the response
172  */
173 public function deleteDog(int $id)
174 {
175     $headers = apache_request_headers();
176
177     if (!isset($headers['Authorization'])) {
178         return ResponseController::notFoundAuthorizationHeader();
179     }
180
181     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
182
183     if (is_null($userAuth) || $userAuth->code_role != Constants::
        ADMIN_CODE_ROLE) {
184         return ResponseController::unauthorizedUser();
185     }
186
187     $dog = $this->DAODog->find($id);
188
189     if (is_null($dog)) {
190         return ResponseController::notFoundResponse();
191     }
192
193     if (!is_null($dog->picture_serial_id)) {
194         $filename = HelperController::getDefaultDirectory(). "storage/app/
            dog_picture/" . $dog->picture_serial_id . ".jpeg";
195         if (file_exists($filename)) {
196             unlink($filename);
197         }
198     }
199
200     $this->DAODog->delete($dog);
201
202     return ResponseController::successfulRequest(null);
203 }
204
205 /**
206  *
207  * Method to upload a dog picture.
208  *
209  * @return string The status and the body in JSON format of the response

```

```

210     */
211     public function uploadDogPicture()
212     {
213         $headers = apache_request_headers();
214
215         if (!isset($headers['Authorization'])) {
216             return ResponseController::notFoundAuthorizationHeader();
217         }
218
219         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
220
221         if (is_null($userAuth) || $userAuth->code_role != Constants::
222             ADMIN_CODE_ROLE) {
223             return ResponseController::unauthorizedUser();
224         }
225
226         if (!isset($_FILES["dog_picture"]) || !is_uploaded_file($_FILES["
227             dog_picture"]["tmp_name"]) || !isset($_POST["dog_id"])) {
228             return ResponseController::unprocessableEntityResponse();
229         }
230
231         $dog = $this->DAODog->find($_POST["dog_id"]);
232
233         if (is_null($dog)) {
234             return ResponseController::notFoundResponse();
235         }
236
237         switch ($_FILES["dog_picture"]["type"]) {
238             case Constants::IMAGE_TYPE_PNG:
239                 HelperController::pngTojpegConverter($_FILES["dog_picture"]["
240                     tmp_name"]);
241                 break;
242             case Constants::IMAGE_TYPE_JPEG:
243                 break;
244             default:
245                 return ResponseController::imageFileFormatProblem();
246                 break;
247         }
248
249         $tmp_file = $_FILES["dog_picture"]["tmp_name"];
250         $img_name = HelperController::generateRandomString();
251         $upload_file = HelperController::getDefaultDirectory(). "storage/app/
252             dog_picture/" . $img_name . ".jpeg";
253
254         if (!is_null($dog->picture_serial_id)) {
255             $filename = HelperController::getDefaultDirectory(). "storage/app/
256                 dog_picture/" . $dog->picture_serial_id . ".jpeg";
257             if (file_exists($filename)) {
258                 unlink($filename);
259             }
260         }
261
262         if (!move_uploaded_file($tmp_file, $upload_file)) {
263             return ResponseController::uploadFailed();
264         }
265
266         $dog->picture_serial_id = $img_name;
267
268         $this->DAODog->update($dog);

```



```

264
265         return ResponseController::successfulRequest();
266     }
267
268     /**
269     *
270     * Method to download a dog picture.
271     *
272     * @param string $serial_id The serial_id of the dog picture
273     * @return string The status and the body in JSON format of the response
274     */
275     public function downloadDogPicture(string $serial_id)
276     {
277         if(is_null($this->DAODog->findBySerialId($serial_id))){
278             return ResponseController::notFoundResponse();
279         }
280
281         $image = HelperController::getDefaultDirectory()."storage/app/dog_picture/"
                . $serial_id.".jpeg";
282
283         $imageInfo = getimagesize($image);
284
285         header("Content-Type: " . $imageInfo["mime"]);
286         header("Content-Length: " . filesize($image));
287         readfile($image);
288
289         return ResponseController::successfulRequest();
290     }
291
292     /**
293     *
294     * Method to check if the dog required fields have been defined for the
        creation.
295     *
296     * @param Dog $dog The dog model object
297     * @return bool
298     */
299     private function validateDog(Dog $dog)
300     {
301         if ($dog->name == null) {
302             return false;
303         }
304
305         if ($dog->breed == null) {
306             return false;
307         }
308
309         if ($dog->sex == null) {
310             return false;
311         }
312
313         if ($dog->user_id == null) {
314             return false;
315         }
316
317         return true;
318     }
319 }

```

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

```

1  <?php
2  /**
3   * HelperController.php
4   *
5   * Controller allowing the use of help functions.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 namespace App\Controllers;
11
12 use PHPMailer\PHPMailer\SMTP;
13 use PHPMailer\PHPMailer\Exception;
14 use PHPMailer\PHPMailer\PHPMailer;
15 use Dompdf\Dompdf;
16
17 class HelperController {
18
19     /**
20     *
21     * Method to check if a date has the right format (YYYY-MM-DD).
22     *
23     * @param string $date Date to check
24     * @return bool
25     */
26     public static function validateDateFormat(string $date)
27     {
28         if (!preg_match("/^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])$/",
29             $date)) {
30             return false;
31         }
32
33         $elements = explode("-", $date);
34
35         if (!checkdate($elements[1], $elements[2], $elements[0])) {
36             return false;
37         }
38
39         return true;
40     }
41
42     /**
43     *
44     * Method to check if a date has the right format (YYYY-MM-DD).
45     *
46     * @param string $date Date to check
47     * @return bool
48     */
49     public static function validateDateTimeFormat(string $datetime, string $format
50         = 'Y-m-d H:i:s')
51     {
52         if (!preg_match("/^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])
53             (0[1-9]|1[0-9]|2[0-4]):(0[0-9]|1[1-5][0-9]):(0[0-9]|1[1-5][0-9])$/",
54             $datetime)) {
55             return false;
56         }
57
58         $date = \DateTime::createFromFormat($format, $datetime);

```

```

55         if ($date && $date->format($format) != $datetime) {
56             return false;
57         }
58         return true;
59     }
60
61     /**
62     *
63     * Method to check if a time has the right format (HH:MM:SS).
64     *
65     * @param string $time Time to check
66     * @return bool
67     */
68     public static function validateTimeFormat(string $time)
69     {
70         if (!preg_match("/^(0[1-9]|1[0-9]|2[0-4]):(0[0-9]|[1-5][0-9])
71             : (0[0-9]|[1-5][0-9])$/",$time)) {
72             return false;
73         }
74         return true;
75     }
76
77     /**
78     *
79     * Method to check if a code day has the right format (1-6).
80     *
81     * @param string $time code day to check
82     * @return bool
83     */
84     public static function validateCodeDayFormat(string $code_day)
85     {
86         if (!preg_match("/^[1-7]$/",$code_day)) {
87             return false;
88         }
89         return true;
90     }
91 }
92
93 /**
94 *
95 * Method to check if the first time is smaller or equal to the second one.
96 *
97 * @param string $firsttime First time to check
98 * @param string $secondtime Second time to check
99 * @return bool
100 */
101 public static function validateChornologicalTime(string $firsttime, string
    $secondtime)
102 {
103     $firstdate = strtotime($firsttime);
104     $seconddate = strtotime($secondtime);
105
106     if ($seconddate < $firstdate) {
107         return false;
108     }
109
110     return true;
111 }

```

```

112
113 /**
114  *
115  * Method to generate an api token.
116  *
117  * @return string The api token
118  */
119 public static function generateApiToken()
120 {
121     return md5(microtime());
122 }
123
124 /**
125  *
126  * Method to generate a random string.
127  *
128  * @return string The random string
129  */
130 public static function generateRandomString() {
131     $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
132     $password = array();
133     $alphaLength = strlen($alphabet) - 1;
134     for ($i = 0; $i < 8; $i++) {
135         $n = rand(0, $alphaLength);
136         $password[] = $alphabet[$n];
137     }
138     return implode($password);
139 }
140
141 /**
142  *
143  * Method to check if a document type has the right value (
144     conditions_inscription,poster).
145  *
146  * @param string $document_type document type to check
147  * @return bool
148  */
149 public static function validateDocumentTypeFormat(string $document_type)
150 {
151     if ($document_type == 'conditions_inscription' || $document_type == 'poster') {
152         return true;
153     }
154     return false;
155 }
156
157 /**
158  *
159  * Method to check if a email has the right format.
160  *
161  * @param string $email email to check
162  * @return bool
163  */
164 public static function validateEmailFormat(string $email)
165 {
166     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
167         return false;

```

```

168     }
169
170     return true;
171 }
172
173 /**
174  *
175  * Method to check if a package number of conditions of registration document
176  * has the right format.
177  *
178  * @param int $package_number Package number to check
179  * @return bool
180  */
181 public static function validatePackageNumber(int $package_number)
182 {
183     if ($package_number < 1 || $package_number > 5) {
184         return false;
185     }
186
187     return true;
188 }
189
190 /**
191  *
192  * Method to load the email HTML CSS template.
193  *
194  * @param string $title The title of the mail
195  * @param string $content The content of the mail
196  * @param string $importantContent The important content of the mail
197  * @return string
198  */
199 public static function loadMailTemplate(string $title, string $content, string
200 $importantContent = null)
201 {
202     $html = '<!DOCTYPE html>
203     <html lang="fr">
204     <head>
205         <meta charset="UTF-8">
206         <meta name="viewport" content="width=device-width,initial-scale=1">
207         <meta name="x-apple-disable-message-reformatting">
208         <title></title>
209         <style>
210             table, td, div, h1, p {font-family: Arial, sans-serif;}
211         </style>
212     </head>
213     <body style="margin:0;padding:0;">
214         <table role="presentation" style="width:100%;border-collapse:collapse;
215         border:0;border-spacing:0;background:#ffffff;">
216             <tr>
217                 <td align="center" style="padding:0;">
218                     <table role="presentation" style="width:602px;border-collapse:
219                     collapse;border:1px solid #cccccc;border-spacing:0;text-align:
220                     left;">

```

```

221     <tr>
222         <td style="padding:36px 30px 42px 30px;">
223             <table role="presentation" style="width:100%;border-collapse:
224                 collapse;border:0;border-spacing:0;">
225                 <tr>
226                     <td style="padding:0 0 36px 0;color:#153643;">
227                         <h1 style="font-size:24px;margin:0 0 20px 0;font-family
228                             :Arial,sans-serif;">'. $title.' </h1>
229                         <p style="text-align: justify;margin:0 0 12px 0;font-
230                             size:16px;line-height:24px;font-family:Arial,sans-
231                             serif;">'. $content.' </p>
232                         <h3 style="font-size:18px;margin:0 0 20px 0;font-family
233                             :Arial,sans-serif;">'. $importantContent.' </h3>
234                     </td>
235                 </tr>
236             </table>
237         </td>
238     </tr>
239     <tr>
240         <td style="padding:30px;background:#3ea3d8;">
241             <table role="presentation" style="width:100%;border-collapse:
242                 collapse;border:0;border-spacing:0;font-size:9px;font-
243                 family:Arial,sans-serif;">
244                 <tr>
245                     <td style="padding:0;width:50%;" align="left">
246                         <p style="margin:0;font-size:14px;line-height:16px;font
247                             -family:Arial,sans-serif;color:#ffffff;">
248                             &reg; Douceur de Chien 2021<br/>
249                             <a href="https://boreljaquet.ch/" style="color:#
250                                 fffffff;text-decoration:underline;">Site web
251                                 Douceur de Chien</a>
252                     </td>
253                     <td style="padding:0;width:50%;" align="right">
254                         <table role="presentation" style="border-collapse:
255                             collapse;border:0;border-spacing:0;">
256                             <tr>
257                                 <td style="padding:0 0 0 10px;width:38px;">
258                                     <a href="http://www.twitter.com/" style="color:#
259                                         fffffff;"></a>
263                                 <td style="padding:0 0 0 10px;width:38px;">
264                                     <a href="http://www.facebook.com/" style="color:#
265                                         fffffff;"></a>
269                                 </td>
270                             </tr>
271                         </table>
272                     </td>
273                 </tr>
274             </table>
275         </td>
276     </tr>
277 </table>

```

```

261         </td>
262     </tr>
263 </table>
264 </body>
265 </html>';
266     return $html;
267 }
268
269 /**
270  *
271  * Method to send an email.
272  *
273  * @param string $message Message of the mail
274  * @param string $subject Subject of the mail
275  * @param string $emailRecipient Recipient's email address
276  * @param string $importantMessage Important message if there is one
277  * @param string $attachmentFilePath Path of the file to attach if there is one
278  * @return void
279  */
280 public static function sendMail(string $message, string $subject, string
    $emailRecipient, string $importantMessage = null, string
    $attachmentFilePath = null)
281 {
282     $mail = new PHPMailer(true);
283     $body = HelperController::loadMailTemplate($subject, $message, $importantMessage)
    ;
284     try {
285         //Server settings
286         $mail->SMTPDebug = SMTP::DEBUG_SERVER;
287         $mail->Host = getenv('SMTP_HOST');
288         $mail->SMTPAuth = true;
289         $mail->Username = getenv('SMTP_USERNAME');
290         $mail->Password = getenv('SMTP_PASSWORD');
291         $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
292         $mail->Port = 587;
293
294         //Recipients
295         $mail->setFrom('noreply@douceurdechien.com', 'Douceur de Chien');
296         $mail->addAddress($emailRecipient);
297
298         //Attachments
299         if (!is_null($attachmentFilePath)) {
300             $mail->addAttachment($attachmentFilePath);
301         }
302
303         //Content
304         $mail->isHTML(true);
305         $mail->CharSet = 'UTF-8';
306         $mail->Encoding = 'base64';
307         $mail->Subject = $subject;
308         $mail->AddEmbeddedImage(HelperController::getDefaultDirectory(). "resources/
            image/logo_douceur_de_chien.png", "logo-image", "logo_douceur_de_chien.
            png");
309         $mail->Body = $body;
310
311         $mail->send();
312     } catch (Exception $e) {
313         echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
314     }

```

```

315     }
316
317     /**
318     *
319     * Method to convert a png image to jpeg image.
320     *
321     * @return void
322     */
323     public static function pngTojpegConverter(string $filename)
324     {
325         $image = imagecreatefrompng ($filename);
326         imagejpeg($image, $filename);
327         imagedestroy($image);
328     }
329
330     /**
331     *
332     * Method to store a conditions of registration.
333     *
334     * @param string $filename The filename of conditions of registration
335     * @param string $package_number The package number of conditions of
336     * registration
337     * @param string $date The date of creation of conditions of registration
338     * @param string $signature_base64 The signature image in base64 of conditions
339     * of registration
340     * @param string $userfirstname The user firstname of owner of conditions of
341     * registration
342     * @param string $userlastname The user lastname of owner of conditions of
343     * registration
344     *
345     * @return void
346     */
347     public static function storeConditionsRegistration(string $filename,int
348         $package_number,string $date, string $signature_base64,string
349         $userfirstname, string $userlastname)
350     {
351         $dompdf = new DOMPDF();
352
353         ob_start();
354         include HelperController::getDefaultDirectory()."resources/template/
355             conditions_registration.php";
356         $contents = ob_get_clean();
357
358         $dompdf->loadHtml($contents);
359         $dompdf->render();
360         $output = $dompdf->output();
361
362         file_put_contents(HelperController::getDefaultDirectory()."storage/app/
363             conditions_registration/".$filename.".pdf", $output);
364     }
365
366     /**
367     *
368     * Method to check if the client-side Google reCAPTCHA is valid or not
369     *
370     * @param string $userResponseToken The user response token provided by the
371     * client-side integration.
372     * @link https://developers.google.com/recaptcha/docs/verify#api-request
373     */

```



```

365     * @return bool
366     */
367     public static function reCAPTCHAvalidate(string $userResponseToken)
368     {
369         $curl = curl_init();
370
371         curl_setopt($curl, CURLOPT_URL, "https://www.google.com/recaptcha/api/
            siteverify");
372         curl_setopt($curl, CURLOPT_POST, 1);
373         curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query(array('secret' =>
            getenv('GOOGLE_RECAPTCHA_SECRET_KEY'), 'response' => $userResponseToken)))
            ;
374         curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
375
376         $response = json_decode(curl_exec($curl));
377
378         return $response->success;
379     }
380
381     /**
382     *
383     * Method to send an email with the appointment information and an ics file (
            iCalendar) as an attachment.
384     *
385     * @param Datetime $start_datetime Start date of the appointment in an Datetime
            instance
386     * @param Datetime $end_datetime End date of the appointment in an Datetime
            instance
387     * @param string $educator_fullname The first and last name of the educator
388     * @param string $customer_email The customer's email address
389     * @param string $filename The name of the ics file to send
390     * @link https://datatracker.ietf.org/doc/html/rfc5545
391     *
392     * @return void
393     */
394     public static function sendMailWithICSFile(\Datetime $start_datetime, \Datetime
            $end_datetime, string $educator_fullname, string $customer_email, string
            $filename)
395     {
396         ob_start();
397         include HelperController::getDefaultDirectory()."resources/template/
            iCalendar_appointment.php";
398         $contents = ob_get_clean();
399
400         $temp = tmpfile();
401         fwrite($temp, $contents);
402
403         $tmpfile_path = stream_get_meta_data($temp)['uri'];
404         $new_tmpfile_path = sys_get_temp_dir(). '\\'. $filename;
405         rename($tmpfile_path, $new_tmpfile_path);
406
407         $message = "Bonjour et merci de faire confiance à la société Douceur de Chien
            ,
408         vous venez de prendre un rendez-vous le ".$start_datetime->format("d-m-Y")."
409         de ".$start_datetime->format("H\\h")." à ".$end_datetime->format("H\\h")." avec
410         l'éducateur canin ".$educator_fullname.".
411         Vous trouverez ci-joint un fichier ICS permettant de planifier le rendez-vous
412         dans votre calendrier Microsoft, Google ou Apple. Vous pouvez également accé
            der aux informations

```

```

413         de ce rendez-vous depuis votre compte.";
414
415         HelperController::sendMail($message,"Un nouveau rendez-vous a été planifié",
            $customer_email,null,$new_tmpfile_path);
416
417         unlink($new_tmpfile_path);
418     }
419
420     /**
421     *
422     * Method to return the default directory of the API.
423     *
424     * @return string The default directory
425     */
426     public static function getDefaultDirectory(){
427         return $_SERVER["DOCUMENT_ROOT"]."/bj-travail-diplome-2021/api_rest/";
428     }
429 }

```

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

```

1  <?php
2  /**
3   * ResponseController.php
4   *
5   * Controller allowing to return different type of HTTP response.
6   *
7   * @link https://developer.mozilla.org/fr/docs/Web/HTTP/Status
8   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
9   */
10
11 namespace App\Controllers;
12
13
14 class ResponseController {
15
16
17
18     /**
19     *
20     * Method to return the error message in case of undefined authorization header
21     *
22     * @return string The status and the body in JSON format of the response
23     */
24     public static function notFoundAuthorizationHeader()
25     {
26         $response['status_code_header'] = 'HTTP/1.1 401 Unauthorized';
27         $response['body'] = json_encode([
28             'error' => "L'en-tête d'autorisation n'est pas défini."
29         ]);
30         return $response;
31     }
32
33     /**
34     *
35     * Method to return the error message in case of no access permission.
36     *
37     * @return string The status and the body in JSON format of the response

```

```

38     */
39     public static function unauthorizedUser()
40     {
41         $response['status_code_header'] = 'HTTP/1.1 403 Forbidden';
42         $response['body'] = json_encode([
43             'error' => "Vous n'avez pas les permissions."
44         ]);
45         return $response;
46     }
47     /**
48      *
49      * Method to return the error message in case of not found response.
50      *
51      * @return string The status and the body in JSON format of the response
52      */
53     public static function notFoundResponse()
54     {
55         $response['status_code_header'] = 'HTTP/1.1 404 Not Found';
56         $response['body'] = json_encode([
57             'error' => "Le serveur n'a pas trouvé la ressource demandée."
58         ]);
59
60         return $response;
61     }
62
63     /**
64      *
65      * Method to return the GET success message.
66      *
67      * @param array $result The associative array containing all the result rows
68      * @return string The status and the body in JSON format of the response
69      */
70     public static function successfulRequest($result = null)
71     {
72         $response = array();
73
74         $response['status_code_header'] = 'HTTP/1.1 200 OK';
75         $response['body'] = (!is_null($result)) ? json_encode($result) : null;
76         return $response;
77     }
78
79     /**
80      *
81      * Method to return the GET success message for data that does not need to be
82      * json encoded.
83      *
84      * @param array $result The associative array containing all the result rows
85      * @return string The status and the body in JSON format of the response
86      */
87     public static function successfulRequestWithoutJson($result)
88     {
89         $response['status_code_header'] = 'HTTP/1.1 200 OK';
90         $response['body'] = $result;
91         return $response;
92     }
93
94     /**
95      *
96      * Method to return the error message in case of invalid attributes.

```

```

96      *
97      * @return string The status and the body in JSON format of the response
98      */
99      public static function unprocessableEntityResponse()
100     {
101         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
102         $response['body'] = json_encode([
103             'error' => 'Attributs invalides.'
104         ]);
105         return $response;
106     }
107
108     /**
109     *
110     * Method to return the error message in case of a permanent calendar already
111     * created.
112     *
113     * @return string The status and the body in JSON format of the response
114     */
115     public static function permanentScheduleAlreadyExist()
116     {
117         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
118         $response['body'] = json_encode([
119             'error' => 'Un calendrier permanent a déjà été créé.'
120         ]);
121         return $response;
122     }
123
124     /**
125     *
126     * Method to return the POST success message.
127     *
128     * @return string The status and the body in JSON format of the response
129     */
130     public static function successfulCreatedRessource()
131     {
132         $response['status_code_header'] = 'HTTP/1.1 201 Created';
133         $response['body'] = null;
134         return $response;
135     }
136
137     /**
138     *
139     * Method to return the POST success message.
140     *
141     * @return string The status and the body in JSON format of the response
142     */
143     public static function successfulCreatedRessourceWithJson($result)
144     {
145         $response['status_code_header'] = 'HTTP/1.1 201 Created';
146         $response['body'] = json_encode($result);
147         return $response;
148     }
149
150     /**
151     *
152     * Method to return the error message in case of invalid date format.
153     *
154     * @return string The status and the body in JSON format of the response

```

```

154     */
155     public static function invalidDateFormat()
156     {
157         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
158         $response['body'] = json_encode([
159             'error' => 'Format de date invalide => (YYYY-MM-DD).'
160         ]);
161         return $response;
162     }
163
164     /**
165      *
166      * Method to return the error message in case of invalid time format.
167      *
168      * @return string The status and the body in JSON format of the response
169      */
170     public static function invalidTimeFormat()
171     {
172         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
173         $response['body'] = json_encode([
174             'error' => 'Format de temps invalide => (HH:MM:SS).'
175         ]);
176         return $response;
177     }
178
179     /**
180      *
181      * Method to return the error message in case of invalid datetime format.
182      *
183      * @return string The status and the body in JSON format of the response
184      */
185     public static function invalidDateTimeFormat()
186     {
187         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
188         $response['body'] = json_encode([
189             'error' => 'Format de date invalide => (YYYY-MM-DD HH:MM:SS).'
190         ]);
191         return $response;
192     }
193
194     /**
195      *
196      * Method to return the error message in case of date overlap problem.
197      *
198      * @return string The status and the body in JSON format of the response
199      */
200     public static function dateOverlapProblem(){
201         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
202         $response['body'] = json_encode([
203             'error' => 'Les dates chevauchent d\'autres dates déjà existantes.'
204         ]);
205         return $response;
206     }
207
208     /**
209      *
210      * Method to return the error message in case of time overlap problem.
211      *
212      * @return string The status and the body in JSON format of the response

```

```

213     */
214     public static function timeOverlapProblem(){
215         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
216         $response['body'] = json_encode([
217             'error' => 'Les horaires chevauchent d\'autres horaires déjà existants.
218         ]);
219         return $response;
220     }
221
222     /**
223      *
224      * Method to return the error message in case of chronological problem between
225      * two times.
226      *
227      * @return string The status and the body in JSON format of the response
228      */
229     public static function chronologicalDateProblem(){
230         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
231         $response['body'] = json_encode([
232             'error' => 'La date ou l\'heure de début est plus récente que la date
233             ou l\'heure de fin.'
234         ]);
235         return $response;
236     }
237
238     /**
239      *
240      * Method to return the error message in case of invalid code day format.
241      *
242      * @return string The status and the body in JSON format of the response
243      */
244     public static function invalidCodeDayFormat(){
245         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
246         $response['body'] = json_encode([
247             'error' => 'Format de jour invalide => (1 jusqu\'à 7, dimanche = 1).'
248         ]);
249         return $response;
250     }
251
252     /**
253      *
254      * Method to return the error message in case of invalid document type format.
255      *
256      * @return string The status and the body in JSON format of the response
257      */
258     public static function invalidDocumentTypeFormat(){
259         $response['status_code_header'] = 'HTTP/1.1 415 Unsupported Media Type';
260         $response['body'] = json_encode([
261             'error' => 'Type de document invalide => (conditions_inscription, poster
262             ).'
263         ]);
264         return $response;
265     }
266
267     /**
268      *
269      * Method to return the error message in case of invalid email format.
270      *

```

```

268     * @return string The status and the body in JSON format of the response
269     */
270     public static function invalidEmailFormat(){
271         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
272         $response['body'] = json_encode([
273             'error' => 'Format d\'adresse email invalide.'
274         ]);
275         return $response;
276     }
277
278     /**
279     *
280     * Method to return the error message in case of invalid login.
281     *
282     * @return string The status and the body in JSON format of the response
283     */
284     public static function invalidLogin(){
285         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
286         $response['body'] = json_encode([
287             'error' => 'Identifiants de connexion invalides.'
288         ]);
289         return $response;
290     }
291
292     /**
293     *
294     * Method to return the error message in case of upload failed.
295     *
296     * @return string The status and the body in JSON format of the response
297     */
298     public static function uploadFailed(){
299         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
300         $response['body'] = json_encode([
301             'error' => 'Échec d\'upload.'
302         ]);
303         return $response;
304     }
305
306     /**
307     *
308     * Method to return the error message in case of invalid image file format.
309     *
310     * @return string The status and the body in JSON format of the response
311     */
312     public static function imageFileFormatProblem(){
313         $response['status_code_header'] = 'HTTP/1.1 415 Unsupported Media Type';
314         $response['body'] = json_encode([
315             'error' => 'Format d\'image par pris en charge.'
316         ]);
317         return $response;
318     }
319
320     /**
321     *
322     * Method to return the error message in case of invalid image file format.
323     *
324     * @return string The status and the body in JSON format of the response
325     */
326     public static function documentTypeNotPdfProblem(){

```

```

327     $response['status_code_header'] = 'HTTP/1.1 415 Unsupported Media Type';
328     $response['body'] = json_encode([
329         'error' => 'Format de document pas pris en charge => Format disponible
                        PDF.'
330     ]);
331     return $response;
332 }
333
334 /**
335  *
336  * Method to return the error message in case of invalid package number of
337  * conditions of registration document.
338  * @return string The status and the body in JSON format of the response
339  */
340 public static function packageNumberFormatProblem(){
341     $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
342     $response['body'] = json_encode([
343         'error' => 'Numéro de Forfait invalide => (1 jusqu\'à 5).'
344     ]);
345     return $response;
346 }
347
348 /**
349  *
350  * Method to return the error message in case of invalid appointment
351  * informations.
352  * @return string The status and the body in JSON format of the response
353  */
354 public static function invalidAppointment(){
355     $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
356     $response['body'] = json_encode([
357         'error' => 'Impossible de prendre ce rendez-vous.'
358     ]);
359     return $response;
360 }
361
362 /**
363  *
364  * Method to return the error message if the email address already exists.
365  *
366  * @return string The status and the body in JSON format of the response
367  */
368 public static function emailAlreadyExist(){
369     $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
370     $response['body'] = json_encode([
371         'error' => 'Un compte avec cette adresse e-mail existe déjà.'
372     ]);
373     return $response;
374 }
375
376 /**
377  *
378  * Method to return the error message in case of invalid captcha.
379  *
380  * @return string The status and the body in JSON format of the response
381  */
382 public static function invalidCaptcha()

```



```

383     {
384         $response['status_code_header'] = 'HTTP/1.1 400 Bad Request';
385         $response['body'] = json_encode([
386             'error' => 'Captcha invalide.'
387         ]);
388         return $response;
389     }
390
391
392 }

```

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

```

1  <?php
2  /**
3   * ScheduleOverrideController.php
4   *
5   * Controller of the ScheduleOverride model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAOScheduleOverride;
13 use App\DataAccessObject\DAOUser;
14 use App\Controllers\ResponseController;
15 use App\Controllers\HelperController;
16 use App\DataAccessObject\DAOTimeSlot;
17 use App\Models\ScheduleOverride;
18 use App\System\Constants;
19
20 class ScheduleOverrideController {
21
22     private DAOScheduleOverride $DAOScheduleOverride;
23     private DAOTimeSlot $DAOTimeSlot;
24     private DAOUser $DAOUser;
25
26     /**
27      *
28      * Constructor of the ScheduleOverrideController object.
29      *
30      * @param PDO $db The database connection
31      */
32     public function __construct(\PDO $db)
33     {
34         $this->DAOScheduleOverride = new DAOScheduleOverride($db);
35         $this->DAOTimeSlot = new DAOTimeSlot($db);
36         $this->DAOUser = new DAOUser($db);
37     }
38
39     /**
40      *
41      * Method to return all schedule overrides and their time slots in JSON format.
42      *
43      * @return string The status and the body in json format of the response
44      */
45     public function getAllScheduleOverrides()
46     {

```

```

47     $headers = apache_request_headers();
48
49     if (!isset($headers['Authorization'])) {
50         return ResponseController::notFoundAuthorizationHeader();
51     }
52
53     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
54
55     if (is_null($userAuth) || $userAuth->code_role != Constants::
56         ADMIN_CODE_ROLE) {
57         return ResponseController::unauthorizedUser();
58     }
59
60     $allScheduleOverrides = $this->DAOScheduleOverride->findAll(false, $userAuth
61         ->id);
62
63     foreach ($allScheduleOverrides as $scheduleOverride) {
64         $scheduleOverride->timeSlots = $this->DAOTimeSlot->
65             findAllByIdScheduleOverride(false, $userAuth->id, $scheduleOverride
66                 ->id);
67     }
68
69     return ResponseController::successfulRequest($allScheduleOverrides);
70 }
71
72 /**
73  *
74  * Method to return a schedule override in JSON format.
75  *
76  * @param int $id The schedule override identifier
77  * @return string The status and the body in JSON format of the response
78  */
79 public function getScheduleOverride(int $id)
80 {
81     $headers = apache_request_headers();
82
83     if (!isset($headers['Authorization'])) {
84         return ResponseController::notFoundAuthorizationHeader();
85     }
86
87     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
88
89     if (is_null($userAuth) || $userAuth->code_role != Constants::
90         ADMIN_CODE_ROLE) {
91         return ResponseController::unauthorizedUser();
92     }
93
94     $scheduleOverride = $this->DAOScheduleOverride->find($id, $userAuth->id);
95
96     if (is_null($scheduleOverride)) {
97         return ResponseController::notFoundResponse();
98     }
99
100    return ResponseController::successfulRequest($scheduleOverride);
101 }
102
103 /**
104  *
105  * Method to create a schedule override.

```

```

101      *
102      * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
103      * @return string The status and the body in JSON format of the response
104      */
105      public function createScheduleOverride(ScheduleOverride $scheduleOverride)
106      {
107          $headers = apache_request_headers();
108
109          if (!isset($headers['Authorization'])) {
110              return ResponseController::notFoundAuthorizationHeader();
111          }
112
113          $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
114
115          if (is_null($userAuth) || $userAuth->code_role != Constants::
116              ADMIN_CODE_ROLE) {
117              return ResponseController::unauthorizedUser();
118          }
119
120          $scheduleOverride->id_educator = $userAuth->id;
121
122          if (!$this->validateScheduleOverride($scheduleOverride)) {
123              return ResponseController::unprocessableEntityResponse();
124          }
125
126          if (!HelperController::validateDateFormat($scheduleOverride->
127              date_schedule_override)) {
128              return ResponseController::invalidDateFormat();
129          }
130
131          if ($this->DAOScheduleOverride->findExistence($scheduleOverride)) {
132              return ResponseController::dateOverlapProblem();
133          }
134
135          $this->DAOScheduleOverride->insert($scheduleOverride);
136
137          return ResponseController::successfulCreatedRessource();
138      }
139
140      /**
141      *
142      * Method to delete a schedule override.
143      *
144      * @param int $id The schedule override identifier
145      * @return string The status and the body in JSON format of the response
146      */
147      public function deleteScheduleOverride(int $id)
148      {
149          $headers = apache_request_headers();
150
151          if (!isset($headers['Authorization'])) {
152              return ResponseController::notFoundAuthorizationHeader();
153          }
154
155          $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
156
157          if (is_null($userAuth) || $userAuth->code_role != Constants::
158              ADMIN_CODE_ROLE) {
159              return ResponseController::unauthorizedUser();
160          }

```

```

157     }
158
159     $scheduleOverride = $this->DAOScheduleOverride->find($id,$userAuth->id);
160
161     if (is_null($scheduleOverride)) {
162         return ResponseController::notFoundResponse();
163     }
164
165     $this->DAOScheduleOverride->delete($scheduleOverride);
166
167     return ResponseController::successfulRequest(null);
168 }
169
170 /**
171  *
172  * Method to check if the schedule override required fields have been defined
173  * for the creation.
174  *
175  * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
176  * @return bool
177  */
178 private function validateScheduleOverride(ScheduleOverride $scheduleOverride)
179 {
180     if ($scheduleOverride->date_schedule_override == null) {
181         return false;
182     }
183
184     return true;
185 }
186 }

```

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

```

1 <?php
2 /**
3  * TimeSlotController.php
4  *
5  * Controller of the TimeSlot model.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAOTimeSlot;
13 use App\DataAccessObject\DAOUser;
14 use App\Controllers\ResponseController;
15 use App\Controllers\HelperController;
16 use App\DataAccessObject\DAOScheduleOverride;
17 use App\DataAccessObject\DAOWeeklySchedule;
18 use App\Models\TimeSlot;
19 use App\System\Constants;
20
21 class TimeSlotController {
22
23     private DAOTimeSlot $DAOTimeSlot;
24     private DAOWeeklySchedule $DAOWeeklySchedule;
25     private DAOScheduleOverride $DAOScheduleOverride;
26     private DAOUser $DAOUser;

```

```

27
28
29 /**
30  *
31  * Constructor of the TimeSlotController object.
32  *
33  * @param PDO $db The database connection
34  */
35 public function __construct(\PDO $db)
36 {
37     $this->DAOTimeSlot = new DAOTimeSlot($db);
38     $this->DAOWeeklySchedule = new DAOWeeklySchedule($db);
39     $this->DAOScheduleOverride = new DAOScheduleOverride($db);
40     $this->DAOUser = new DAOUser($db);
41 }
42
43 /**
44  *
45  * Method to return all time slots in JSON format.
46  *
47  * @return string The status and the body in json format of the response
48  */
49 public function getAllTimeSlots()
50 {
51     $headers = apache_request_headers();
52
53     if (!isset($headers['Authorization'])) {
54         return ResponseController::notFoundAuthorizationHeader();
55     }
56
57     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
58
59     if (is_null($userAuth) || $userAuth->code_role != Constants::
60         ADMIN_CODE_ROLE) {
61         return ResponseController::unauthorizedUser();
62     }
63
64     $allTimeSlots = $this->DAOTimeSlot->findAll(false, $userAuth->id);
65
66     return ResponseController::successfulRequest($allTimeSlots);
67 }
68
69 /**
70  *
71  * Method to return a time slot in JSON format.
72  *
73  * @param int $id The time slot identifier
74  * @return string The status and the body in JSON format of the response
75  */
76 public function getTimeSlot(int $id)
77 {
78     $headers = apache_request_headers();
79
80     if (!isset($headers['Authorization'])) {
81         return ResponseController::notFoundAuthorizationHeader();
82     }
83
84     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);

```

```

85         if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
86             return ResponseController::unauthorizedUser();
87         }
88
89         $timeSlot = $this->DAOTimeSlot->find($id,$userAuth->id);
90
91         if (is_null($timeSlot)) {
92             return ResponseController::notFoundResponse();
93         }
94
95         return ResponseController::successfulRequest($timeSlot);
96     }
97
98     /**
99     *
100     * Method to create a time slot.
101     *
102     * @param TimeSlot $timeSlot The timeSlot model object
103     * @return string The status and the body in JSON format of the response
104     */
105     public function createTimeSlot(TimeSlot $timeSlot)
106     {
107         $headers = apache_request_headers();
108
109         if (!isset($headers['Authorization'])) {
110             return ResponseController::notFoundAuthorizationHeader();
111         }
112
113         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
114
115         if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
116             return ResponseController::unauthorizedUser();
117         }
118
119         $timeSlot->id_educator = $userAuth->id;
120
121         if (!$this->validateTimeSlot($timeSlot)) {
122             return ResponseController::unprocessableEntityResponse();
123         }
124
125         if (!HelperController::validateCodeDayFormat($timeSlot->code_day)) {
126             return ResponseController::invalidCodeDayFormat();
127         }
128
129         if (!HelperController::validateTimeFormat($timeSlot->time_start) || !
            HelperController::validateTimeFormat($timeSlot->time_end) ) {
130             return ResponseController::invalidTimeFormat();
131         }
132
133         if (!HelperController::validateChornologicalTime($timeSlot->time_start ,
            $timeSlot->time_end)) {
134             return ResponseController::chronologicalDateProblem();
135         }
136
137         if ($this->DAOTimeSlot->findOverlapInWeeklySchedule($timeSlot) || $this->
            DAOTimeSlot->findOverlapInScheduleOverride($timeSlot))
138         {

```

```

139         return ResponseController::timeOverlapProblem();
140     }
141
142     $this->DAOTimeSlot->insert($timeSlot);
143
144     return ResponseController::successfulCreatedResource();
145 }
146
147 /**
148  *
149  * Method to delete a time slot.
150  *
151  * @param int $id The time slot identifier
152  * @return string The status and the body in JSON format of the response
153  */
154 public function deleteTimeSlot(int $id)
155 {
156     $headers = apache_request_headers();
157
158     if (!isset($headers['Authorization'])) {
159         return ResponseController::notFoundAuthorizationHeader();
160     }
161
162     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
163
164     if (is_null($userAuth) || $userAuth->code_role != Constants::
        ADMIN_CODE_ROLE) {
165         return ResponseController::unauthorizedUser();
166     }
167
168     $timeSlot = $this->DAOTimeSlot->find($id, $userAuth->id);
169
170     if (is_null($timeSlot)) {
171         return ResponseController::notFoundResponse();
172     }
173
174     $this->DAOTimeSlot->delete($timeSlot);
175
176     return ResponseController::successfulRequest(null);
177 }
178
179 /**
180  *
181  * Method to return all available valid time slots with their corresponding
    dates in JSON format.
182  *
183  * @param int $idUser The educator user identifier
184  * @return string The status and the body in json format of the response
185  */
186 public function getPlanningTimeSlots(int $idUser)
187 {
188     $educator = $this->DAOUser->find($idUser);
189
190     if (is_null($educator) || $educator->code_role != Constants::
        ADMIN_CODE_ROLE) {
191         return ResponseController::notFoundResponse();
192     }
193
194     $planning = $this->DAOTimeSlot->findPlanningForEducator($idUser);

```

```

195         return ResponseController::successfulRequest($planning);
196     }
197
198     /**
199     *
200     * Method to check if the time slot required fields have been defined for the
201     * creation.
202     *
203     * @param TimeSlot $timeSlot The timeslot model object
204     * @return bool
205     */
206     private function validateTimeSlot(TimeSlot $timeSlot)
207     {
208         if ($timeSlot->code_day == null) {
209             return false;
210         }
211
212         if ($timeSlot->time_start == null) {
213             return false;
214         }
215
216         if ($timeSlot->time_end == null) {
217             return false;
218         }
219
220         if (($timeSlot->id_weekly_schedule != null) xor ($timeSlot->
221             id_schedule_override == null)) {
222             return false;
223         }
224
225         $result = null;
226
227         if ($timeSlot->id_weekly_schedule != null) {
228             $result = $this->DAOWeeklySchedule->find($timeSlot->id_weekly_schedule,
229                 $timeSlot->id_educator);
230         }
231
232         if ($timeSlot->id_schedule_override != null) {
233             $result = $this->DAOScheduleOverride->find($timeSlot->
234                 id_schedule_override, $timeSlot->id_educator);
235         }
236
237         if ($result == null) {
238             return false;
239         }
240     }

```

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

```

1 <?php
2 /**
3  * UserController.php
4  *
5  * Controller of the User model.
6  *

```



```

7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAOUser;
13 use App\DataAccessObject\DAODog;
14 use App\DataAccessObject\DAODocument;
15 use App\DataAccessObject\DAOAppointment;
16 use app\Models\User;
17 use App\Controllers\ResponseController;
18 use App\System\Constants;
19
20 class UserController {
21
22     private DAOUser $DAOUser;
23     private DAODog $DAODog;
24     private DAODocument $DAODocument;
25     private DAOAppointment $DAOAppointment;
26
27     /**
28      *
29      * Constructor of the UserController object.
30      *
31      * @param PDO $db The database connection
32      */
33     public function __construct(\PDO $db)
34     {
35         $this->DAOUser = new DAOUser($db);
36         $this->DAODog = new DAODog($db);
37         $this->DAODocument = new DAODocument($db);
38         $this->DAOAppointment = new DAOAppointment($db);
39     }
40
41     /**
42      *
43      * Method to return all users in JSON format.
44      *
45      * @return string The status and the body in json format of the response
46      */
47     public function getAllCustomerUsers()
48     {
49         $headers = apache_request_headers();
50
51         if (!isset($headers['Authorization'])) {
52             return ResponseController::notFoundAuthorizationHeader();
53         }
54
55         $api_token = $headers['Authorization'];
56
57         $userAuth = $this->DAOUser->findByApiToken($api_token);
58
59         if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
            return ResponseController::unauthorizedUser();
60         }
61
62         $allCustomerUsers = $this->DAOUser->findAll(Constants::USER_CODE_ROLE);
63
64

```

```

65         foreach ($allCustomerUsers as $customerUser) {
66             $customerUser->dogs = $this->DAODog->findByUserId($customerUser->id);
67         }
68
69         return ResponseController::successfulRequest($allCustomerUsers);
70     }
71
72     /**
73      *
74      * Method to return all educator users in JSON format.
75      *
76      * @return string The status and the body in json format of the response
77      */
78     public function getAllEducatorUsers()
79     {
80         $allEducatorUsers = $this->DAOUser->findAllEducators();
81
82         return ResponseController::successfulRequest($allEducatorUsers);
83     }
84
85     /**
86      *
87      * Method to return a user in JSON format.
88      *
89      * @param int $id The user identifier
90      * @return string The status and the body in JSON format of the response
91      */
92     public function getUser(int $id)
93     {
94         $headers = apache_request_headers();
95
96         if (!isset($headers['Authorization'])) {
97             return ResponseController::notFoundAuthorizationHeader();
98         }
99
100         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
101
102         if (is_null($userAuth) || $userAuth->code_role != Constants::
            ADMIN_CODE_ROLE) {
103             return ResponseController::unauthorizedUser();
104         }
105
106         $user = $this->DAOUser->find($id);
107
108         if (is_null($user)) {
109             return ResponseController::notFoundResponse();
110         }
111
112         $dogs = $this->DAODog->findByUserId($user->id);
113         $documents = $this->DAODocument->findByUserId($user->id);
114         $appointments = $this->DAOAppointment->findByUserIdForAdmin($user->id);
115         $user->dogs = $dogs;
116         $user->documents = $documents;
117         $user->appointments = $appointments;
118
119         return ResponseController::successfulRequest($user);
120     }
121
122     /**

```

```

123      *
124      * Method to create a user.
125      *
126      * @param User $user The user model object
127      * @param string $reCAPTCHAuserResponseToken The user response token provided
        by the reCAPTCHA client-side integration.
128      * @return string The status and the body in JSON format of the response
129      */
130      public function createUser(User $user, string $reCAPTCHAuserResponseToken =
        null)
131      {
132          $user->api_token = HelperController::generateApiToken();
133          $user->code_role = Constants::USER_CODE_ROLE;
134
135          if (!$this->validateUser($user)) {
136              return ResponseController::unprocessableEntityResponse();
137          }
138
139          if (!HelperController::validateEmailFormat($user->email)) {
140              return ResponseController::invalidEmailFormat();
141          }
142
143          if (!is_null($this->DAOUser->findUserByEmail($user->email))) {
144              return ResponseController::emailAlreadyExist();
145          }
146
147          if ($user->password_hash != null) {
148              $user->password_hash = password_hash($user->password_hash,
                PASSWORD_DEFAULT);
149              if (is_null($reCAPTCHAuserResponseToken)) {
150                  return ResponseController::unprocessableEntityResponse();
151              }
152              if (!HelperController::reCAPTCHAvalidate($reCAPTCHAuserResponseToken))
                {
153                  return ResponseController::invalidCaptcha();
154              }
155          }
156          else{
157              $random_password = HelperController::generateRandomString();
158              $user->password_hash = password_hash($random_password,PASSWORD_DEFAULT)
                ;
159          }
160
161          $this->DAOUser->insert($user);
162
163          if (isset($random_password)) {
164              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);
165          }
166
167          $newUser = $this->DAOUser->findUserByEmail($user->email);
168
169          $result = array();
170          $result["api_token"] = $user->api_token;
171          $result["user_id"] = $newUser->id;

```

```

172     $result["code_role"] = $user->code_role;
173     return ResponseController::successfulCreatedRessourceWithJson($result);
174 }
175
176 /**
177  *
178  * Method to update a user.
179  *
180  * @param User $user The user model object
181  * @return string The status and the body in JSON format of the response
182  */
183 public function updateUser(User $user)
184 {
185     $headers = apache_request_headers();
186
187     if (!isset($headers['Authorization'])) {
188         return ResponseController::notFoundAuthorizationHeader();
189     }
190
191     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
192
193     if (is_null($userAuth) || $userAuth->code_role != Constants::
194         ADMIN_CODE_ROLE) {
195         return ResponseController::unauthorizedUser();
196     }
197
198     $actualUser = $this->DAOUser->find($user->id);
199
200     if (is_null($actualUser)) {
201         return ResponseController::notFoundResponse();
202     }
203
204     $actualUser->email = $user->email ?? $actualUser->email;
205     $actualUser->firstname = $user->firstname ?? $actualUser->firstname;
206     $actualUser->lastname = $user->lastname ?? $actualUser->lastname;
207     $actualUser->phonenumber = $user->phonenumber ?? $actualUser->phonenumber;
208     $actualUser->address = $user->address ?? $actualUser->address;
209
210     if (!HelperController::validateEmailFormat($actualUser->email)) {
211         return ResponseController::invalidEmailFormat();
212     }
213
214     $this->DAOUser->update($actualUser);
215
216     return ResponseController::successfulRequest(null);
217 }
218
219 /**
220  *
221  * Method to delete a user.
222  *
223  * @param int $id The user identifier
224  * @return string The status and the body in JSON format of the response
225  */
226 public function deleteUser(int $id)
227 {
228     $headers = apache_request_headers();
229
230     if (!isset($headers['Authorization'])) {

```

```

230         return ResponseController::notFoundAuthorizationHeader();
231     }
232
233     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
234
235     if (is_null($userAuth) || $userAuth->code_role != Constants::
        ADMIN_CODE_ROLE) {
236         return ResponseController::unauthorizedUser();
237     }
238
239     $user = $this->DAOUser->find($id);
240
241     if (is_null($user)) {
242         return ResponseController::notFoundResponse();
243     }
244
245     $dogs = $this->DAODog->findByUserId($user->id);
246
247     foreach ($dogs as $dog) {
248
249         if (!is_null($dog->picture_serial_id)) {
250
251             $filename = HelperController::getDefaultDirectory(). "storage/app/
                dog_picture/" . $dog->picture_serial_id . ".jpeg";
252
253             if (file_exists($filename)) {
254                 unlink($filename);
255             }
256         }
257     }
258
259     $documents = $this->DAODocument->findByUserId($user->id);
260
261     foreach ($documents as $document) {
262
263         if (!is_null($document->document_serial_id)) {
264
265             if ($document->type == Constants::
                DOCUMENT_TYPE_CONDITIONS_OF_REGISTRATION) {
266
267                 $filename = HelperController::getDefaultDirectory(). "storage/
                    app/conditions_registration/" . $document->document_serial_id
                    . ".pdf";
268
269                 }
270                 else{
271                     $filename = HelperController::getDefaultDirectory(). "storage/
                        app/pdf/" . $document->document_serial_id . ".pdf";
272                 }
273
274                 if (file_exists($filename)) {
275                     unlink($filename);
276                 }
277             }
278         }
279
280     $appointments = $this->DAOAppointment->findByUserId($user->id);
281
282     foreach ($appointments as $appointment) {

```

```

283
284         if (!is_null($appointment->note_graphical_serial_id)) {
285
286             $filename = HelperController::getDefaultDirectory()."storage/app/
graphical_note/".$appointment->note_graphical_serial_id.".png";
287
288             if (file_exists($filename)) {
289                 unlink($filename);
290             }
291         }
292     }
293
294     $this->DAOUser->delete($user);
295
296     return ResponseController::successfulRequest(null);
297 }
298
299 /**
300  *
301  * Method to get the api token of a user by logging in.
302  *
303  * @param User $user The user model object
304  * @return string The status and the body in JSON format of the response
305  */
306 public function connection(User $user)
307 {
308     if (is_null($user->email) || is_null($user->password_hash)) {
309         return ResponseController::unprocessableEntityResponse();
310     }
311
312     $userAuth = $this->DAOUser->findUserByEmail($user->email);
313
314     if (is_null($userAuth)) {
315         return ResponseController::invalidLogin();
316     }
317
318     if (!password_verify($user->password_hash, $userAuth->password_hash)) {
319         return ResponseController::invalidLogin();
320     }
321
322     $userAuth->api_token = HelperController::generateApiToken();
323
324     $this->DAOUser->update($userAuth);
325
326     $result = array();
327     $result["api_token"] = $userAuth->api_token;
328     $result["user_id"] = $userAuth->id;
329     $result["code_role"] = $userAuth->code_role;
330     return ResponseController::successfulRequest($result);
331 }
332
333 /**
334  *
335  * Method to return all information of the currently logged in user in JSON
format.
336  *
337  * @return string The status and the body in JSON format of the response
338  */
339 public function getMyInformations()

```

```

340 {
341     $headers = apache_request_headers();
342
343     if (!isset($headers['Authorization'])) {
344         return ResponseController::notFoundAuthorizationHeader();
345     }
346
347     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
348
349     if (is_null($userAuth)) {
350         return ResponseController::notFoundResponse();
351     }
352     $dogs = $this->DAODog->findByUserId($userAuth->id);
353     $documents = $this->DAODocument->findByUserId($userAuth->id);
354     $appointments = $this->DAOAppointment->findByUserId($userAuth->id);
355     $userAuth->dogs = $dogs;
356     $userAuth->documents = $documents;
357     $userAuth->appointments = $appointments;
358
359
360     return ResponseController::successfulRequest($userAuth);
361 }
362
363 /**
364  *
365  * Method to update the password of the currently logged user.
366  *
367  * @param User $user The user model object
368  * @return string The status and the body in JSON format of the response
369  */
370 public function updateMyPassword(User $user)
371 {
372     $headers = apache_request_headers();
373
374     if (!isset($headers['Authorization'])) {
375         return ResponseController::notFoundAuthorizationHeader();
376     }
377
378     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
379
380     if (is_null($userAuth)) {
381         return ResponseController::notFoundResponse();
382     }
383
384     $userAuth->password_hash = password_hash($user->password_hash,
385         PASSWORD_DEFAULT) ?? $userAuth->password_hash;
386
387     $this->DAOUser->update($userAuth);
388
389     return ResponseController::successfulRequest(null);
390 }
391
392 /**
393  *
394  * Method to check if the user required fields have been defined for the
395  * creation.
396  *
397  * @param User $user The user model object
398  * @return bool

```

```

397     */
398     private function validateUser(User $user)
399     {
400         if ($user->email == null) {
401             return false;
402         }
403
404         if ($user->firstname == null) {
405             return false;
406         }
407
408         if ($user->lastname == null) {
409             return false;
410         }
411
412         if ($user->phonenummer == null) {
413             return false;
414         }
415
416         if ($user->address == null) {
417             return false;
418         }
419
420         return true;
421     }
422 }

```

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

```

1  <?php
2  /**
3   * WeeklyScheduleController.php
4   *
5   * Controller of the WeeklySchedule model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 namespace App\Controllers;
11
12 use App\DataAccessObject\DAOWeeklySchedule;
13 use App\DataAccessObject\DAOUser;
14 use App\Controllers\ResponseController;
15 use App\Controllers\HelperController;
16 use App\DataAccessObject\DAOTimeSlot;
17 use App\Models\WeeklySchedule;
18 use App\System\Constants;
19
20 class WeeklyScheduleController {
21
22     private DAOWeeklySchedule $DAOWeeklySchedule;
23     private DAOTimeSlot $DAOTimeSlot;
24     private DAOUser $DAOUser;
25
26     /**
27      *
28      * Constructor of the WeeklyScheduleController object.
29      *
30      * @param PDO $db The database connection

```



```

31     */
32 public function __construct(\PDO $db)
33 {
34     $this->DAOWeeklySchedule = new DAOWeeklySchedule($db);
35     $this->DAOTimeSlot = new DAOTimeSlot($db);
36     $this->DAOUser = new DAOUser($db);
37 }
38
39 /**
40  *
41  * Method to return all weekly schedules and their time slots in JSON format.
42  *
43  * @return string The status and the body in json format of the response
44  */
45 public function getAllWeeklySchedules()
46 {
47     $headers = apache_request_headers();
48
49     if (!isset($headers['Authorization'])) {
50         return ResponseController::notFoundAuthorizationHeader();
51     }
52
53     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
54
55     if (is_null($userAuth) || $userAuth->code_role != Constants::
56         ADMIN_CODE_ROLE) {
57         return ResponseController::unauthorizedUser();
58     }
59
60     $allWeeklySchedule = $this->DAOWeeklySchedule->findAll(false, $userAuth->id)
61         ;
62
63     foreach ($allWeeklySchedule as $weeklySchedule) {
64         $weeklySchedule->timeSlots = $this->DAOTimeSlot->
65             findAllByIdWeeklySchedule(false, $userAuth->id, $weeklySchedule->id
66             );
67     }
68
69     return ResponseController::successfulRequest($allWeeklySchedule);
70 }
71
72 /**
73  *
74  * Method to return a weekly schedule in JSON format.
75  *
76  * @param int $id The weekly schedule identifier
77  * @return string The status and the body in JSON format of the response
78  */
79 public function getWeeklySchedule(int $id)
80 {
81     $headers = apache_request_headers();
82
83     if (!isset($headers['Authorization'])) {
84         return ResponseController::notFoundAuthorizationHeader();
85     }
86
87     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
88
89     if (is_null($userAuth) || $userAuth->code_role != Constants::

```

```

86         ADMIN_CODE_ROLE) {
87             return ResponseController::unauthorizedUser();
88         }
89
90         $weeklySchedule = $this->DAOWeeklySchedule->find($id,$userAuth->id);
91
92         if (is_null($weeklySchedule)) {
93             return ResponseController::notFoundResponse();
94         }
95
96         return ResponseController::successfulRequest($weeklySchedule);
97     }
98
99     /**
100      *
101      * Method to create a weekly schedule.
102      * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
103      * @return string The status and the body in JSON format of the response
104      */
105     public function createWeeklySchedule(WeeklySchedule $weeklySchedule)
106     {
107         $headers = apache_request_headers();
108
109         if (!isset($headers['Authorization'])) {
110             return ResponseController::notFoundAuthorizationHeader();
111         }
112
113         $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
114
115         if (is_null($userAuth) || $userAuth->code_role != Constants::
116             ADMIN_CODE_ROLE) {
117             return ResponseController::unauthorizedUser();
118         }
119
120         $weeklySchedule->id_educator = $userAuth->id;
121
122         if (!$this->validateWeeklySchedule($weeklySchedule)) {
123             return ResponseController::unprocessableEntityResponse();
124         }
125
126         if (!HelperController::validateDateFormat($weeklySchedule->date_valid_from)
127             ) {
128             return ResponseController::invalidDateFormat();
129         }
130
131         if (!is_null($weeklySchedule->date_valid_to)) {
132             if (!HelperController::validateDateFormat($weeklySchedule->
133                 date_valid_to)) {
134                 return ResponseController::invalidDateFormat();
135             }
136             if (!HelperController::validateChornologicalTime($weeklySchedule->
137                 date_valid_from,$weeklySchedule->date_valid_to)) {
138                 return ResponseController::chronologicalDateProblem();
139             }
140         }
141
142         else{
143             if ($this->DAOWeeklySchedule->findActifPermanentSchedule(
144                 $weeklySchedule)) {

```

```

139         return ResponseController::permanentScheduleAlreadyExist();
140     }
141 }
142
143 if ($this->DAOWeeklySchedule->findOverlap($weeklySchedule)) {
144     return ResponseController::dateOverlapProblem();
145 }
146
147 $this->DAOWeeklySchedule->insert($weeklySchedule);
148
149 return ResponseController::successfulCreatedRessource();
150 }
151
152 /**
153  *
154  * Method to delete a weekly schedule.
155  *
156  * @param int $id The weekly schedule identifier
157  * @return string The status and the body in JSON format of the response
158  */
159 public function deleteWeeklySchedule(int $id)
160 {
161     $headers = apache_request_headers();
162
163     if (!isset($headers['Authorization'])) {
164         return ResponseController::notFoundAuthorizationHeader();
165     }
166
167     $userAuth = $this->DAOUser->findByApiToken($headers['Authorization']);
168
169     if (is_null($userAuth) || $userAuth->code_role != Constants::
170         ADMIN_CODE_ROLE) {
171         return ResponseController::unauthorizedUser();
172     }
173
174     $weeklySchedule = $this->DAOWeeklySchedule->find($id,$userAuth->id);
175
176     if (is_null($weeklySchedule)) {
177         return ResponseController::notFoundResponse();
178     }
179
180     $this->DAOWeeklySchedule->delete($weeklySchedule);
181
182     return ResponseController::successfulRequest(null);
183 }
184
185 /**
186  *
187  * Method to check if the weekly schedule required fields have been defined for
188  * the creation.
189  *
190  * @param WeeklySchedule $weeklySchedule The weeklyschedul model object
191  * @return bool
192  */
193 private function validateWeeklySchedule(WeeklySchedule $weeklySchedule)
194 {
195     if ($weeklySchedule->date_valid_from == null) {
196         return false;
197     }
198 }

```

```

196
197         return true;
198     }
199 }

```

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

```

1  <?php
2  /**
3   * DAOAbsence.php
4   *
5   * Data access object of the absence table.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\DataAccessObject;
10
11  use App\Models\Absence;
12
13  class DAOAbsence {
14
15      private $db = null;
16
17      /**
18       *
19       * Constructor of the DAOAbsence object.
20       *
21       * @param PDO $db The database connection
22       */
23      public function __construct(\PDO $db)
24      {
25          $this->db = $db;
26      }
27
28      /**
29       *
30       * Method to return all the absences of the database in an array of absence
31       * objects.
32       *
33       * @param bool $isDeleted Bool to define whether to search for existing or
34       * deleted absences
35       * @param int $idEducator The educator identifier
36       * @return Absence[] A Absence object array
37       */
38      public function findAll(bool $isDeleted, int $idEducator)
39      {
40          $statement = "
41          SELECT id, date_absence_from, date_absence_to, description, id_educator
42          FROM absence
43          WHERE is_deleted= :DELETED
44          AND id_educator = :ID_EDUCATOR
45          ORDER BY date_absence_from";
46
47          try {
48              $statement = $this->db->prepare($statement);
49              $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
50              $statement->bindParam(':DELETED', $isDeleted, \PDO::PARAM_BOOL);
51              $statement->execute();
52              $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

```

```

51
52     $absenceArray = array();
53
54     foreach ($results as $result) {
55         $absence = new Absence();
56         $absence->id = $result["id"];
57         $absence->date_absence_from = $result["date_absence_from"];
58         $absence->date_absence_to = $result["date_absence_to"];
59         $absence->description = $result["description"];
60         $absence->id_educator = $result["id_educator"];
61         array_push($absenceArray, $absence);
62     }
63
64     return $absenceArray;
65 } catch (\PDOException $e) {
66     exit($e->getMessage());
67 }
68 }
69
70 /**
71  *
72  * Method to return an absence from the database in a absence model object.
73  *
74  * @param int $id The absence identifier
75  * @param int $idEducator The educator identifier
76  * @return Absence A Absence model object containing all the result rows of the
77  * query
78 */
79 public function find(int $id, int $idEducator)
80 {
81     $statement = "
82     SELECT id, date_absence_from, date_absence_to, description, id_educator
83     FROM absence
84     WHERE id = :ID_ABSENCE
85     AND is_deleted = 0
86     AND id_educator = :ID_EDUCATOR";
87
88     try {
89         $statement = $this->db->prepare($statement);
90         $statement->bindParam(':ID_ABSENCE', $id, \PDO::PARAM_INT);
91         $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
92         $statement->execute();
93
94         $absence = new Absence();
95
96         if ($statement->rowCount() == 1) {
97             $result = $statement->fetch(\PDO::FETCH_ASSOC);
98             $absence->id = $result["id"];
99             $absence->date_absence_from = $result["date_absence_from"];
100             $absence->date_absence_to = $result["date_absence_to"];
101             $absence->description = $result["description"];
102             $absence->id_educator = $result["id_educator"];
103         }
104         else {
105             $absence = null;
106         }
107
108         return $absence;
109     } catch (\PDOException $e) {

```

```

109         exit($e->getMessage());
110     }
111 }
112
113 /**
114  *
115  * Method to insert a absence in the database.
116  *
117  * @param Absence $absence The absence model object
118  * @return int The number of rows affected by the insert
119  */
120 public function insert(Absence $absence)
121 {
122     $statement = "
123     INSERT INTO absence (date_absence_from, date_absence_to, description,
124         id_educator ,is_deleted)
125     VALUES (STR_TO_DATE(:DATE_ABSENCE_FROM, \"%Y-%m-%d\"), STR_TO_DATE(:
126         DATE_ABSENCE_TO, \"%Y-%m-%d\"), :DESCRIPTION,:ID_EDUCATOR ,0)";
127
128     try {
129         $statement = $this->db->prepare($statement);
130         $statement->bindParam(':DATE_ABSENCE_FROM', $absence->date_absence_from
131             , \PDO::PARAM_STR);
132         $statement->bindParam(':DATE_ABSENCE_TO', $absence->date_absence_to, \
133             PDO::PARAM_STR);
134         $statement->bindParam(':DESCRIPTION', $absence->description, \PDO::
135             PARAM_STR);
136         $statement->bindParam(':ID_EDUCATOR', $absence->id_educator, \PDO::
137             PARAM_INT);
138         $statement->execute();
139         return $statement->rowCount();
140     } catch (\PDOException $e) {
141         exit($e->getMessage());
142     }
143 }
144
145 /**
146  *
147  * Method to update a absence in the database.
148  *
149  * @param Absence $absence The absence model object
150  * @return int The number of rows affected by the update
151  */
152 public function update(Absence $absence)
153 {
154     $statement = "
155     UPDATE absence
156     SET date_absence_from = STR_TO_DATE(:DATE_ABSENCE_FROM, \"%Y-%m-%d\"),
157     date_absence_to = STR_TO_DATE(:DATE_ABSENCE_TO, \"%Y-%m-%d\"),
158     description = :DESCRIPTION
159     WHERE id = :ID_ABSENCE";
160
161     try {
162         $statement = $this->db->prepare($statement);
163         $statement->bindParam(':DATE_ABSENCE_FROM', $absence->date_absence_from
164             , \PDO::PARAM_STR);
165         $statement->bindParam(':DATE_ABSENCE_TO', $absence->date_absence_to, \
166             PDO::PARAM_STR);
167         $statement->bindParam(':DESCRIPTION', $absence->description, \PDO::

```

```

        PARAM_STR);
160     $statement->bindParam(':ID_ABSENCE', $absence->id, \PDO::PARAM_INT);
161     $statement->execute();
162     return $statement->rowCount();
163 } catch (\PDOException $e) {
164     exit($e->getMessage());
165 }
166 }
167
168 /**
169  *
170  * Method to delete a absence in the database.
171  *
172  * @param Absence $absence The absence model object
173  * @return int The number of rows affected by the delete
174  */
175 public function delete(Absence $absence)
176 {
177     $statement = "
178     UPDATE absence
179     SET is_deleted = 1
180     WHERE id = :ID_ABSENCE";
181
182     try {
183         $statement = $this->db->prepare($statement);
184         $statement->bindParam(':ID_ABSENCE', $absence->id, \PDO::PARAM_INT);
185         $statement->execute();
186         return $statement->rowCount();
187     } catch (\PDOException $e) {
188         exit($e->getMessage());
189     }
190 }
191 }

```

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

```

1 <?php
2 /**
3  * DAOAppointment.php
4  *
5  * Data access object of the appointment table.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\DataAccessObject;
10
11 use App\Models\Appointment;
12
13 class DAOAppointment {
14
15     private $db = null;
16
17     /**
18      *
19      * Constructor of the DAOAppointment object.
20      *
21      * @param PDO $db The database connection
22      */
23     public function __construct(\PDO $db)

```

```

24 {
25     $this->db = $db;
26 }
27
28 /**
29  *
30  * Method to return all the appointments of the database in an array of
    appointment objects.
31  *
32  * @param int $userCustomerId The customer identifier
33  * @param int $userEducatorId The educator identifier
34  * @return Appointment[] A Appointment object array
35  */
36 public function findAll(int $userCustomerId = null, int $userEducatorId = null)
37 {
38     $statement = "
39     SELECT id, datetime_appointment,duration_in_hour, note_text,
40     note_graphical_serial_id, summary, datetime_deletion,
41     user_id_customer,user_id_educator,user_id_deletion
42     FROM appointment
43     WHERE (user_id_customer = :ID_USER_CUSTOMER OR user_id_educator = :
44           ID_USER_EDUCATOR)
45     AND datetime_deletion IS NULL
46     AND user_id_deletion IS NULL";
47
48     try {
49         $statement = $this->db->prepare($statement);
50         $statement->bindParam(':ID_USER_CUSTOMER', $userCustomerId, \PDO::
51             PARAM_INT);
52         $statement->bindParam(':ID_USER_EDUCATOR', $userEducatorId, \PDO::
53             PARAM_INT);
54         $statement->execute();
55         $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
56
57         $appointmentArray = array();
58
59         foreach ($results as $result) {
60             $appointment = new Appointment();
61             $appointment->id = $result["id"];
62             $appointment->datetime_appointment = $result["datetime_appointment"]
63             ];
64             $appointment->duration_in_hour = $result["duration_in_hour"];
65             $appointment->note_text = $result["note_text"];
66             $appointment->note_graphical_serial_id = $result["
67                 note_graphical_serial_id"];
68             $appointment->summary = $result["summary"];
69             $appointment->user_id_customer = $result["user_id_customer"];
70             $appointment->user_id_educator = $result["user_id_educator"];
71             array_push($appointmentArray,$appointment);
72         }
73
74         return $appointmentArray;
75     } catch (\PDOException $e) {
76         exit($e->getMessage());
77     }
78 }
79
80 /**
81  *

```



```

77      * Method to return a appointment slot from the database in a appointment model
       object.
78      *
79      * @param int $id The appointment identifier
80      * @param int $idEducator The educator identifier
81      * @return Appointment A Appointment model object containing all the result
       rows of the query
82      */
83      public function find(int $id)
84      {
85          $statement = "
86          SELECT id, datetime_appointment,duration_in_hour, note_text,
87          note_graphical_serial_id, summary, datetime_deletion,
88          user_id_customer,user_id_educator,user_id_deletion
89          FROM appointment
90          WHERE id = :ID_APPOINTMENT";
91
92          try {
93              $statement = $this->db->prepare($statement);
94              $statement->bindParam(':ID_APPOINTMENT', $id, \PDO::PARAM_INT);
95              $statement->execute();
96
97              $appointment = new Appointment();
98
99              if ($statement->rowCount()==1) {
100                  $result = $statement->fetch(\PDO::FETCH_ASSOC);
101                  $appointment->id = $result["id"];
102                  $appointment->datetime_appointment = $result["datetime_appointment"
103                  ];
104                  $appointment->duration_in_hour = $result["duration_in_hour"];
105                  $appointment->note_text = $result["note_text"];
106                  $appointment->note_graphical_serial_id = $result["
107                      note_graphical_serial_id"];
108                  $appointment->summary = $result["summary"];
109                  $appointment->user_id_customer = $result["user_id_customer"];
110                  $appointment->user_id_educator = $result["user_id_educator"];
111              }
112              else{
113                  $appointment = null;
114              }
115
116              return $appointment;
117          } catch (\PDOException $e) {
118              exit($e->getMessage());
119          }
120      }
121
122      /**
123      *
124      * Method to return all appointment from the database with the user id in an
125      array of appointment objects without the administrator data.
126      *
127      * @param int $userId The user identifier
128      * @return Appointment[] A Appointment object array
129      */
130      public function findByUserId(int $userId)
131      {
132          $statement = "
133          SELECT id, datetime_appointment,duration_in_hour, summary,

```

```

131         user_id_customer,user_id_educator
132     FROM appointment
133     WHERE user_id_customer = :ID_USER
134     AND datetime_deletion IS NULL
135     AND user_id_deletion IS NULL
136     ORDER BY datetime_appointment DESC";
137
138     try {
139         $statement = $this->db->prepare($statement);
140         $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
141         $statement->execute();
142         $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
143
144         $appointmentArray = array();
145
146         foreach ($results as $result) {
147             $appointment = new Appointment();
148             $appointment->id = $result["id"];
149             $appointment->datetime_appointment = $result["datetime_appointment"];
150
151             $appointment->duration_in_hour = $result["duration_in_hour"];
152             $appointment->summary = $result["summary"];
153             $appointment->user_id_customer = $result["user_id_customer"];
154             $appointment->user_id_educator = $result["user_id_educator"];
155             array_push($appointmentArray,$appointment);
156         }
157
158         return $appointmentArray;
159     } catch (\PDOException $e) {
160         exit($e->getMessage());
161     }
162 }
163
164 /**
165  * Method to return all appointment from the database with the user id in an
166  * array of appointment objects.
167  *
168  * @param int $userId The user identifier
169  * @return Appointment[] A Appointment object array
170  */
171 public function findByUserIdForAdmin(int $userId)
172 {
173     $statement = "
174     SELECT id, datetime_appointment,duration_in_hour,note_text,
175             note_graphical_serial_id, summary,
176     user_id_customer,user_id_educator
177     FROM appointment
178     WHERE user_id_customer = :ID_USER
179     AND datetime_deletion IS NULL
180     AND user_id_deletion IS NULL
181     ORDER BY datetime_appointment DESC";
182
183     try {
184         $statement = $this->db->prepare($statement);
185         $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
186         $statement->execute();
187         $results = $statement->fetchAll(\PDO::FETCH_ASSOC);

```

```

187         $appointmentArray = array();
188
189         foreach ($results as $result) {
190             $appointment = new Appointment();
191             $appointment->id = $result["id"];
192             $appointment->datetime_appointment = $result["datetime_appointment"];
193             $appointment->duration_in_hour = $result["duration_in_hour"];
194             $appointment->note_text = $result["note_text"];
195             $appointment->note_graphical_serial_id = $result["
                note_graphical_serial_id"];
196             $appointment->summary = $result["summary"];
197             $appointment->user_id_customer = $result["user_id_customer"];
198             $appointment->user_id_educator = $result["user_id_educator"];
199             array_push($appointmentArray, $appointment);
200         }
201
202         return $appointmentArray;
203     } catch (\PDOException $e) {
204         exit($e->getMessage());
205     }
206 }
207
208 /**
209  *
210  * Method to return a appointment from the database from his graphical note
    serial id.
211  *
212  * @param string $serial_id The graphical note serial id
213  * @return Appointment A Appointment model object containing all the result
    rows of the query
214  */
215 public function findBySerialId(string $serial_id)
216 {
217     $statement = "
218     SELECT id, datetime_appointment, duration_in_hour, note_text,
219     note_graphical_serial_id, summary, datetime_deletion,
220     user_id_customer, user_id_educator, user_id_deletion
221     FROM appointment
222     WHERE note_graphical_serial_id = :SERIAL_ID";
223
224     try {
225         $statement = $this->db->prepare($statement);
226         $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
227         $statement->execute();
228
229         $appointment = new Appointment();
230
231         if ($statement->rowCount() == 1) {
232             $result = $statement->fetch(\PDO::FETCH_ASSOC);
233             $appointment->id = $result["id"];
234             $appointment->datetime_appointment = $result["datetime_appointment"];
235             $appointment->duration_in_hour = $result["duration_in_hour"];
236             $appointment->note_text = $result["note_text"];
237             $appointment->note_graphical_serial_id = $result["
                note_graphical_serial_id"];
238             $appointment->summary = $result["summary"];
239             $appointment->user_id_customer = $result["user_id_customer"];

```

```

240         $appointment->user_id_educator = $result["user_id_educator"];
241     }
242     else{
243         $appointment = null;
244     }
245
246     return $appointment;
247
248 } catch (\PDOException $e) {
249     exit($e->getMessage());
250 }
251 }
252
253 /**
254  *
255  * Method to insert a appointment in the database.
256  *
257  * @param Appointment $appointment The absence model object
258  * @return int The number of rows affected by the insert
259  */
260 public function insert(Appointment $appointment)
261 {
262     $statement = "
263     INSERT INTO appointment (datetime_appointment,duration_in_hour,
264     user_id_customer,user_id_educator)
265
266     VALUES (:DATETIME_APPOINTMENT, :DURATION_IN_HOUR,
267     :USER_ID_CUSTOMER, :USER_ID_EDUCATOR)";
268
269     try {
270         $statement = $this->db->prepare($statement);
271         $statement->bindParam(':DATETIME_APPOINTMENT', $appointment->
272             datetime_appointment, \PDO::PARAM_STR);
273         $statement->bindParam(':DURATION_IN_HOUR', $appointment->
274             duration_in_hour, \PDO::PARAM_INT);
275         $statement->bindParam(':USER_ID_CUSTOMER', $appointment->
276             user_id_customer, \PDO::PARAM_INT);
277         $statement->bindParam(':USER_ID_EDUCATOR', $appointment->
278             user_id_educator, \PDO::PARAM_INT);
279         $statement->execute();
280         return $statement->rowCount();
281     } catch (\PDOException $e) {
282         exit($e->getMessage());
283     }
284 }
285
286 /**
287  *
288  * Method to update a appointment in the database.
289  *
290  * @param Appointment $appointment The absence model object
291  * @return int The number of rows affected by the update
292  */
293 public function update(Appointment $appointment)
294 {
295     $statement = "
296     UPDATE appointment
297     SET note_text = :NOTE_TEXT,
298     note_graphical_serial_id = :NOTE_GRAPHICAL_SERIAL_ID,

```

```

295         summary = :SUMMARY
296         WHERE id = :ID_APPOINTMENT";
297
298     try {
299         $statement = $this->db->prepare($statement);
300         $statement->bindParam(':NOTE_TEXT', $appointment->note_text, \PDO::
            PARAM_STR);
301         $statement->bindParam(':NOTE_GRAPHICAL_SERIAL_ID', $appointment->
            note_graphical_serial_id, \PDO::PARAM_STR);
302         $statement->bindParam(':SUMMARY', $appointment->summary, \PDO::
            PARAM_STR);
303         $statement->bindParam(':ID_APPOINTMENT', $appointment->id, \PDO::
            PARAM_INT);
304         $statement->execute();
305         return $statement->rowCount();
306     } catch (\PDOException $e) {
307         exit($e->getMessage());
308     }
309 }
310
311 /**
312  *
313  * Method to delete a appointment in the database.
314  *
315  * @param int $userId The user who deletes the appointment
316  * @param int $id The appointment identifier
317  * @return int The number of rows affected by the delete
318  */
319 public function delete(int $id,int $idUser)
320 {
321     $statement = "
322     UPDATE appointment
323     SET datetime_deletion = NOW(),
324     user_id_deletion = :ID_USER
325     WHERE id = :ID_APPOINTMENT";
326
327     try {
328         $statement = $this->db->prepare($statement);
329         $statement->bindParam(':ID_USER', $idUser, \PDO::PARAM_INT);
330         $statement->bindParam(':ID_APPOINTMENT', $id, \PDO::PARAM_INT);
331         $statement->execute();
332         return $statement->rowCount();
333     } catch (\PDOException $e) {
334         exit($e->getMessage());
335     }
336 }
337 }

```

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

```

1 <?php
2 /**
3  * DAODocument.php
4  *
5  * Data access object of the document table.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\DataAccessObject;

```

```

10
11 use App\Models\Document;
12
13 class DAODocument {
14
15     private \PDO $db;
16
17     /**
18      *
19      * Constructor of the DAODocument object.
20      *
21      * @param PDO $db The database connection
22      */
23     public function __construct(\PDO $db)
24     {
25         $this->db = $db;
26     }
27
28     /**
29      *
30      * Method to return all the documents of the database in an array of document
31      * objects.
32      *
33      * @return Document[] A Document object array
34      */
35     public function findAll()
36     {
37         $statement = "
38         SELECT id, document_serial_id, type, user_id
39         FROM document";
40
41         try {
42             $statement = $this->db->prepare($statement);
43             $statement->execute();
44             $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
45
46             $documentArray = array();
47
48             foreach ($results as $result) {
49                 $document = new Document();
50                 $document->id = $result["id"];
51                 $document->document_serial_id = $result["document_serial_id"];
52                 $document->type = $result["type"];
53                 $document->user_id = $result["user_id"];
54                 array_push($documentArray, $document);
55             }
56
57             return $documentArray;
58         } catch (\PDOException $e) {
59             exit($e->getMessage());
60         }
61     }
62
63     /**
64      *
65      * Method to return a document from the database in a document model object.
66      *
67      * @param int $id The document identifier

```

```

68      * @return Document A Document model object containing all the result rows of
        the query
69      */
70      public function find(int $id)
71      {
72          $statement = "
73          SELECT id, document_serial_id, type, user_id
74          FROM document
75          WHERE id = :ID_DOCUMENT";
76
77          try {
78              $statement = $this->db->prepare($statement);
79              $statement->bindParam(':ID_DOCUMENT', $id, \PDO::PARAM_INT);
80              $statement->execute();
81
82              $document = new Document();
83
84              if ($statement->rowCount()==1) {
85                  $result = $statement->fetch(\PDO::FETCH_ASSOC);
86                  $document->id = $result["id"];
87                  $document->document_serial_id = $result["document_serial_id"];
88                  $document->type = $result["type"];
89                  $document->user_id = $result["user_id"];
90              }
91              else{
92                  $document = null;
93              }
94
95              return $document;
96          } catch (\PDOException $e) {
97              exit($e->getMessage());
98          }
99      }
100
101      /**
102      *
103      * Method to return all documents from the database with the user id in an
        array of documents objects.
104      *
105      * @param int $userId The user identifier
106      * @return Document[] A Document object array
107      */
108      public function findByUserId(int $userId)
109      {
110          $statement = "
111          SELECT id, document_serial_id, type, user_id
112          FROM document
113          WHERE user_id = :ID_USER";
114
115          try {
116              $statement = $this->db->prepare($statement);
117              $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
118              $statement->execute();
119              $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
120
121              $documentArray = array();
122
123              foreach ($results as $result) {
124                  $document = new Document();

```

```

125         $document->id = $result["id"];
126         $document->document_serial_id = $result["document_serial_id"];
127         $document->type = $result["type"];
128         $document->user_id = $result["user_id"];
129         array_push($documentArray,$document);
130     }
131
132     return $documentArray;
133
134     } catch (\PDOException $e) {
135         exit($e->getMessage());
136     }
137 }
138
139 /**
140  *
141  * Method to return a document from the database with the serial id in a
142     document model object.
143  *
144  * @param string $serial_id The serial id of the document
145  * @return Document A Document model object containing all the result rows of
146     the query
147  */
148 public function findBySerialId(string $serial_id)
149 {
150     $statement = "
151     SELECT id, document_serial_id, type, user_id
152     FROM document
153     WHERE document_serial_id= :SERIAL_ID";
154
155     try {
156         $statement = $this->db->prepare($statement);
157         $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
158         $statement->execute();
159
160         $document = new Document();
161
162         if ($statement->rowCount()==1) {
163             $result = $statement->fetch(\PDO::FETCH_ASSOC);
164             $document->id = $result["id"];
165             $document->document_serial_id = $result["document_serial_id"];
166             $document->type = $result["type"];
167             $document->user_id = $result["user_id"];
168         }
169         else{
170             $document = null;
171         }
172
173         return $document;
174
175     } catch (\PDOException $e) {
176         exit($e->getMessage());
177     }
178 }
179
180 /**
181  *
182  * Method to return a document from the database with the user id and serial id
183     in a document model object.

```



```

181      *
182      * @param int $userId The user identifier
183      * @param string $serial_id The serial id of the document
184      * @return Document A Document model object containing all the result rows of
        the query
185      */
186      public function findByUserIdAndSerialId(int $userId,string $serial_id)
187      {
188          $statement = "
189          SELECT id, document_serial_id, type, user_id
190          FROM document
191          WHERE user_id = :ID_USER
192          AND document_serial_id= :SERIAL_ID";
193
194          try {
195              $statement = $this->db->prepare($statement);
196              $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
197              $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
198              $statement->execute();
199
200              $document = new Document();
201
202              if ($statement->rowCount()==1) {
203                  $result = $statement->fetch(\PDO::FETCH_ASSOC);
204                  $document->id = $result["id"];
205                  $document->document_serial_id = $result["document_serial_id"];
206                  $document->type = $result["type"];
207                  $document->user_id = $result["user_id"];
208              }
209              else{
210                  $document = null;
211              }
212
213              return $document;
214
215          } catch (\PDOException $e) {
216              exit($e->getMessage());
217          }
218      }
219
220      /**
221      *
222      * Method to insert a document in the database.
223      *
224      * @param Document $document The document model object
225      * @return int The number of rows affected by the insert
226      */
227      public function insert(Document $document)
228      {
229          $statement = "
230          INSERT INTO document (document_serial_id, type,user_id)
231          VALUES (:DOCUMENT_SERIAL_ID, :TYPE, :USER_ID)";
232
233          try {
234              $statement = $this->db->prepare($statement);
235              $statement->bindParam(':DOCUMENT_SERIAL_ID', $document->
                document_serial_id, \PDO::PARAM_STR);
236              $statement->bindParam(':TYPE', $document->type, \PDO::PARAM_STR);
237              $statement->bindParam(':USER_ID', $document->user_id, \PDO::PARAM_STR);

```

```

238         $statement->execute();
239         return $statement->rowCount();
240     } catch (\PDOException $e) {
241         exit($e->getMessage());
242     }
243 }
244
245 /**
246  *
247  * Method to update a document in the database.
248  *
249  * @param Document $document The document model object
250  * @return int The number of rows affected by the update
251  */
252 public function update(Document $document)
253 {
254     $statement = "
255     UPDATE document
256     SET document_serial_id = :DOCUMENT_SERIAL_ID,
257     type = :TYPE,
258     user_id = :USER_ID
259     WHERE id = :ID_DOCUMENT";
260
261     try {
262         $statement = $this->db->prepare($statement);
263         $statement->bindParam(':DOCUMENT_SERIAL_ID', $document->
            document_serial_id, \PDO::PARAM_STR);
264         $statement->bindParam(':TYPE', $document->type, \PDO::PARAM_STR);
265         $statement->bindParam(':USER_ID', $document->user_id, \PDO::PARAM_STR);
266         $statement->bindParam(':ID_DOCUMENT', $document->id, \PDO::PARAM_INT);
267         $statement->execute();
268         return $statement->rowCount();
269     } catch (\PDOException $e) {
270         exit($e->getMessage());
271     }
272 }
273
274 /**
275  *
276  * Method to delete a document in the database.
277  *
278  * @param Document $document The document model object
279  * @return int The number of rows affected by the delete
280  */
281 public function delete(Document $document)
282 {
283     $statement = "
284     DELETE FROM document
285     WHERE id = :ID_DOCUMENT";
286
287     try {
288         $statement = $this->db->prepare($statement);
289         $statement->bindParam(':ID_DOCUMENT', $document->id, \PDO::PARAM_INT);
290         $statement->execute();
291         return $statement->rowCount();
292     } catch (\PDOException $e) {
293         exit($e->getMessage());
294     }
295 }

```

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

```

1  <?php
2  /**
3   * DAODog.php
4   *
5   * Data access object of the dog table.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\DataAccessObject;
10
11  use App\Models\Dog;
12
13
14  class DAODog {
15
16      private \PDO $db;
17
18      /**
19       *
20       * Constructor of the DAODog object.
21       *
22       * @param PDO $db The database connection
23       */
24      public function __construct(\PDO $db)
25      {
26          $this->db = $db;
27      }
28
29      /**
30       *
31       * Method to return all dogs from the database in an array of dog objects.
32       *
33       * @return Dog[] A Dog object array
34       */
35      public function findAll()
36      {
37          $statement = "
38          SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
39          FROM dog";
40
41          try {
42              $statement = $this->db->prepare($statement);
43              $statement->execute();
44              $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
45
46              $dogArray = array();
47
48              foreach ($results as $result) {
49                  $dog = new Dog();
50                  $dog->id = $result["id"];
51                  $dog->name = $result["name"];
52                  $dog->breed = $result["breed"];
53                  $dog->sex = $result["sex"];
54                  $dog->picture_serial_id = $result["picture_serial_id"];
55                  $dog->chip_id = $result["chip_id"];

```

```

56         $dog->user_id = $result["user_id"];
57         array_push($dogArray,$dog);
58     }
59
60     return $dogArray;
61
62     } catch (\PDOException $e) {
63         exit($e->getMessage());
64     }
65 }
66
67
68 /**
69  *
70  * Method to return a dog from the database in a dog model object.
71  *
72  * @param int $id The dog identifier
73  * @return Dog A Dog model object containing all the result rows of the query
74  */
75 public function find(int $id)
76 {
77     $statement = "
78     SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
79     FROM dog
80     WHERE id = :ID_DOG";
81
82     try {
83         $statement = $this->db->prepare($statement);
84         $statement->bindParam(':ID_DOG', $id, \PDO::PARAM_INT);
85         $statement->execute();
86
87         $dog = new Dog();
88
89         if ($statement->rowCount()==1) {
90             $result = $statement->fetch(\PDO::FETCH_ASSOC);
91             $dog->id = $result["id"];
92             $dog->name = $result["name"];
93             $dog->breed = $result["breed"];
94             $dog->sex = $result["sex"];
95             $dog->picture_serial_id = $result["picture_serial_id"];
96             $dog->chip_id = $result["chip_id"];
97             $dog->user_id = $result["user_id"];
98         }
99         else{
100             $dog = null;
101         }
102
103         return $dog;
104     } catch (\PDOException $e) {
105         exit($e->getMessage());
106     }
107 }
108
109 /**
110  *
111  * Method to return all dogs from the database with the user id in an array of
112  * dog objects.
113  *
114  * @param int $userId The user identifier

```

```

114     * @return Dog[] A Dog object array
115     */
116     public function findByUserId(int $userId)
117     {
118         $statement = "
119         SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
120         FROM dog
121         WHERE user_id = :ID_USER";
122
123         try {
124             $statement = $this->db->prepare($statement);
125             $statement->bindParam(':ID_USER', $userId, \PDO::PARAM_INT);
126             $statement->execute();
127             $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
128
129             $dogArray = array();
130
131             foreach ($results as $result) {
132                 $dog = new Dog();
133                 $dog->id = $result["id"];
134                 $dog->name = $result["name"];
135                 $dog->breed = $result["breed"];
136                 $dog->sex = $result["sex"];
137                 $dog->picture_serial_id = $result["picture_serial_id"];
138                 $dog->chip_id = $result["chip_id"];
139                 $dog->user_id = $result["user_id"];
140                 array_push($dogArray, $dog);
141             }
142
143             return $dogArray;
144
145         } catch (\PDOException $e) {
146             exit($e->getMessage());
147         }
148     }
149
150     /**
151     *
152     * Method to return a dog from the database from his picture serial id.
153     *
154     * @param string $serial_id The dog picture serial id
155     * @return Dog A Dog model object containing all the result rows of the query
156     */
157     public function findBySerialId(string $serial_id)
158     {
159         $statement = "
160         SELECT id, name, breed, sex, picture_serial_id, chip_id, user_id
161         FROM dog
162         WHERE picture_serial_id = :SERIAL_ID";
163
164         try {
165             $statement = $this->db->prepare($statement);
166             $statement->bindParam(':SERIAL_ID', $serial_id, \PDO::PARAM_STR);
167             $statement->execute();
168
169             $dog = new Dog();
170
171             if ($statement->rowCount() == 1) {
172                 $result = $statement->fetch(\PDO::FETCH_ASSOC);

```

```

173         $dog->id = $result["id"];
174         $dog->name = $result["name"];
175         $dog->breed = $result["breed"];
176         $dog->sex = $result["sex"];
177         $dog->picture_serial_id = $result["picture_serial_id"];
178         $dog->chip_id = $result["chip_id"];
179         $dog->user_id = $result["user_id"];
180     }
181     else{
182         $dog = null;
183     }
184
185     return $dog;
186
187     } catch (\PDOException $e) {
188         exit($e->getMessage());
189     }
190 }
191
192 /**
193  *
194  * Method to insert a dog in the database.
195  *
196  * @param Dog $dog The dog model object
197  * @return int The number of rows affected by the insert
198  */
199 public function insert(Dog $dog)
200 {
201     $statement = "
202     INSERT INTO dog (name, breed, sex, picture_serial_id, chip_id, user_id)
203     VALUES (:NAME, :BREED, :SEX, :PICTURE_SERIAL_ID, :CHIP_ID, :USER_ID)";
204
205     try {
206         $statement = $this->db->prepare($statement);
207         $statement->bindParam(':NAME', $dog->name, \PDO::PARAM_STR);
208         $statement->bindParam(':BREED', $dog->breed, \PDO::PARAM_STR);
209         $statement->bindParam(':SEX', $dog->sex, \PDO::PARAM_STR);
210         $statement->bindParam(':PICTURE_SERIAL_ID', $dog->picture_serial_id, \
211             PDO::PARAM_STR);
212         $statement->bindParam(':CHIP_ID', $dog->chip_id, \PDO::PARAM_STR);
213         $statement->bindParam(':USER_ID', $dog->user_id, \PDO::PARAM_STR);
214         $statement->execute();
215         return $statement->rowCount();
216     } catch (\PDOException $e) {
217         exit($e->getMessage());
218     }
219 }
220
221 /**
222  *
223  * Method to update a dog in the database.
224  *
225  * @param Dog $dog The dog model object
226  * @return int The number of rows affected by the update
227  */
228 public function update(Dog $dog)
229 {
230     $statement = "
231     UPDATE dog

```

```

231         SET name = :NAME,
232         breed = :BREED,
233         sex = :SEX,
234         picture_serial_id = :PICTURE_SERIAL_ID,
235         chip_id = :CHIP_ID
236         WHERE id = :ID_DOG";
237
238     try {
239         $statement = $this->db->prepare($statement);
240         $statement->bindParam(':NAME', $dog->name, \PDO::PARAM_STR);
241         $statement->bindParam(':BREED', $dog->breed, \PDO::PARAM_STR);
242         $statement->bindParam(':SEX', $dog->sex, \PDO::PARAM_STR);
243         $statement->bindParam(':PICTURE_SERIAL_ID', $dog->picture_serial_id, \
                PDO::PARAM_STR);
244         $statement->bindParam(':CHIP_ID', $dog->chip_id, \PDO::PARAM_STR);
245         $statement->bindParam(':ID_DOG', $dog->id, \PDO::PARAM_INT);
246         $statement->execute();
247         return $statement->rowCount();
248     } catch (\PDOException $e) {
249         exit($e->getMessage());
250     }
251 }
252
253 /**
254  *
255  * Method to delete a dog in the database.
256  *
257  * @param Dog $dog The dog model object
258  * @return int The number of rows affected by the delete
259  */
260 public function delete(Dog $dog)
261 {
262     $statement = "
263     DELETE FROM dog
264     WHERE id = :ID_DOG";
265
266     try {
267         $statement = $this->db->prepare($statement);
268         $statement->bindParam(':ID_DOG', $dog->id, \PDO::PARAM_INT);
269         $statement->execute();
270         return $statement->rowCount();
271     } catch (\PDOException $e) {
272         exit($e->getMessage());
273     }
274 }
275 }

```

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

```

1 <?php
2 /**
3  * DAOScheduleOverride.php
4  *
5  * Data access object of the schedule_override table.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\DataAccessObject;
10

```

```

11 use App\Models\ScheduleOverride;
12
13 class DAOScheduleOverride {
14
15     private $db = null;
16
17     /**
18      *
19      * Constructor of the DAOScheduleOverride object.
20      *
21      * @param PDO $db The database connection
22      */
23     public function __construct(\PDO $db)
24     {
25         $this->db = $db;
26     }
27
28     /**
29      *
30      * Method to return all the schedule override of the database in an array of
31      * scheduleoverride objects.
32      *
33      * @param bool $isDeleted Bool to define whether to search for existing or
34      * deleted scheduleoverride
35      * @param int $idEducator The educator identifier
36      * @return ScheduleOverride[] A ScheduleOverride object array
37      */
38     public function findAll(bool $deleted, int $idEducator)
39     {
40         $statement = "
41         SELECT id, date_schedule_override, id_educator
42         FROM schedule_override
43         WHERE is_deleted= :DELETED
44         AND id_educator = :ID_EDUCATOR
45         ORDER BY date_schedule_override";
46
47         try {
48             $statement = $this->db->prepare($statement);
49             $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
50             $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
51             $statement->execute();
52             $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
53
54             $scheduleOverrideArray = array();
55
56             foreach ($results as $result) {
57                 $scheduleOverride = new ScheduleOverride();
58                 $scheduleOverride->id = $result["id"];
59                 $scheduleOverride->date_schedule_override = $result["
60                     date_schedule_override"];
61                 $scheduleOverride->id_educator = $result["id_educator"];
62                 array_push($scheduleOverrideArray, $scheduleOverride);
63             }
64
65             return $scheduleOverrideArray;
66         } catch (\PDOException $e) {
67             exit($e->getMessage());
68         }
69     }
70 }

```



```

67
68 /**
69 *
70 * Method to return a schedule override from the database in a scheduleoverride
    model object.
71 *
72 * @param int $id The scheduleoverride identifier
73 * @param int $idEducator The educator identifier
74 * @return ScheduleOverride A ScheduleOverride model object containing all the
    result rows of the query
75 */
76 public function find(int $id, int $idEducator)
77 {
78     $statement = "
79     SELECT id, date_schedule_override, id_educator
80     FROM schedule_override
81     WHERE id = :ID_SCHEDULE_OVERRIDE
82     AND is_deleted = 0
83     AND id_educator = :ID_EDUCATOR";
84
85     try {
86         $statement = $this->db->prepare($statement);
87         $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $id, \PDO::PARAM_INT);
88         $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
89         $statement->execute();
90
91         $scheduleOverride = new ScheduleOverride();
92
93         if ($statement->rowCount()==1) {
94             $result = $statement->fetch(\PDO::FETCH_ASSOC);
95             $scheduleOverride->id = $result["id"];
96             $scheduleOverride->date_schedule_override = $result["
                date_schedule_override"];
97             $scheduleOverride->id_educator = $result["id_educator"];
98         }
99         else{
100             $scheduleOverride = null;
101         }
102
103         return $scheduleOverride;
104     } catch (\PDOException $e) {
105         exit($e->getMessage());
106     }
107 }
108
109 /**
110 *
111 * Method to insert a schedule override in the database.
112 *
113 * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
114 * @return int The number of rows affected by the insert
115 */
116 public function insert(ScheduleOverride $scheduleOverride)
117 {
118     $statement = "
119     INSERT INTO schedule_override (date_schedule_override, id_educator,
        is_deleted)
120     VALUES (STR_TO_DATE(:DATE_SCHEDULE_OVERRIDE, \"%Y-%m-%d\"), :ID_EDUCATOR, 0)
        ";

```

```

121
122     try {
123         $statement = $this->db->prepare($statement);
124         $statement->bindParam(':DATE_SCHEDULE_OVERRIDE', $scheduleOverride->
            date_schedule_override, \PDO::PARAM_STR);
125         $statement->bindParam(':ID_EDUCATOR', $scheduleOverride->id_educator, \
            PDO::PARAM_INT);
126         $statement->execute();
127         return $statement->rowCount();
128     } catch (\PDOException $e) {
129         exit($e->getMessage());
130     }
131 }
132
133 /**
134  *
135  * Method to update a schedule override in the database.
136  *
137  * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
138  * @return int The number of rows affected by the update
139  */
140 public function update(ScheduleOverride $scheduleOverride)
141 {
142     $statement = "
143     UPDATE schedule_override
144     SET date_schedule_override = STR_TO_DATE(:DATE_SCHEDULE_OVERRIDE, \"%Y-%m-%
        d\")
145     WHERE id = :ID_SCHEDULE_OVERRIDE";
146
147     try {
148         $statement = $this->db->prepare($statement);
149         $statement->bindParam(':DATE_SCHEDULE_OVERRIDE', $scheduleOverride->
            date_schedule_override, \PDO::PARAM_STR);
150         $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $scheduleOverride->id, \
            PDO::PARAM_INT);
151         $statement->execute();
152         return $statement->rowCount();
153     } catch (\PDOException $e) {
154         exit($e->getMessage());
155     }
156 }
157
158 /**
159  *
160  * Method to delete a schedule override in the database.
161  *
162  * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
163  * @return int The number of rows affected by the delete
164  */
165 public function delete(ScheduleOverride $scheduleOverride)
166 {
167     $statement = "
168     UPDATE schedule_override
169     SET is_deleted = 1
170     WHERE id = :ID_SCHEDULE_OVERRIDE";
171
172     try {
173         $statement = $this->db->prepare($statement);
174         $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $scheduleOverride->id, \

```

```

175         PDO::PARAM_INT);
176         $statement->execute();
177         return $statement->rowCount();
178     } catch (\PDOException $e) {
179         exit($e->getMessage());
180     }
181 }
182
183 /**
184  * Method to check if a date has not already been defined for the same educator
185  * in the database.
186  *
187  * @param ScheduleOverride $scheduleOverride The scheduleoverride model object
188  * @return array The associative array containing all the result rows of the
189  * query
190  */
191 public function findExistence(ScheduleOverride $scheduleOverride)
192 {
193     $statement = "
194     SELECT *
195     FROM schedule_override
196     WHERE date_schedule_override = STR_TO_DATE(:DATE_SCHEDULE_OVERRIDE, \"%Y-%m
197           -%d\")
198     AND is_deleted = 0
199     AND id_educator = :ID_EDUCATOR";
200
201     try {
202         $statement = $this->db->prepare($statement);
203         $statement->bindParam(':DATE_SCHEDULE_OVERRIDE', $scheduleOverride->
204             date_schedule_override, \PDO::PARAM_STR);
205         $statement->bindParam(':ID_EDUCATOR', $scheduleOverride->id_educator, \
206             PDO::PARAM_INT);
207         $statement->execute();
208         $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
209         return $result;
210     } catch (\PDOException $e) {
211         exit($e->getMessage());
212     }
213 }
214 }

```

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

```

1 <?php
2 /**
3  * DAOTimeSlot.php
4  *
5  * Data access object of the time_slot table.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\DataAccessObject;
10
11 use App\Models\TimeSlot;
12
13 class DAOTimeSlot {
14
15     private $db = null;

```

```

16
17 /**
18 *
19 * Constructor of the DAOTimeSlot object.
20 *
21 * @param PDO $db The database connection
22 */
23 public function __construct(\PDO $db)
24 {
25     $this->db = $db;
26 }
27
28 /**
29 *
30 * Method to return all the time slot for an educator of the database in an
    array of timeslot objects.
31 *
32 * @param bool $isDeleted Bool to define whether to search for existing or
    deleted timeslot
33 * @param int $idEducator The educator identifier
34 * @return TimeSlot[] A ScheduleOverride object array
35 */
36 public function findAll(bool $deleted,int $idEducator)
37 {
38     $statement = "
39     SELECT id, code_day, time_start, time_end,id_weekly_schedule,
        id_schedule_override,id_educator
40     FROM time_slot
41     WHERE is_deleted= :DELETED
42     AND id_educator = :ID_EDUCATOR";
43
44     try {
45         $statement = $this->db->prepare($statement);
46         $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
47         $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
48         $statement->execute();
49         $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
50
51         $timeSlotArray = array();
52
53         foreach ($results as $result) {
54             $timeSlot = new TimeSlot();
55             $timeSlot->id = $result["id"];
56             $timeSlot->code_day = $result["code_day"];
57             $timeSlot->time_start = $result["time_start"];
58             $timeSlot->time_end = $result["time_end"];
59             $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
60             $timeSlot->id_schedule_override = $result["id_schedule_override"];
61             $timeSlot->id_educator = $result["id_educator"];
62             array_push($timeSlotArray,$timeSlot);
63         }
64
65         return $timeSlotArray;
66     } catch (\PDOException $e) {
67         exit($e->getMessage());
68     }
69 }
70
71 /**

```

```

72  *
73  * Method to return all the time slot of the database in an array of timeslot
    objects from his weekly schedule identifier.
74  *
75  * @param bool $isDeleted Bool to define whether to search for existing or
    deleted timeslot
76  * @param int $idEducator The educator identifier
77  * @param int $idWeeklySchedule The weekly schedule identifier
78  * @return TimeSlot[] A ScheduleOverride object array
79  */
80  public function findAllByIdWeeklySchedule(bool $deleted,int $idEducator, int
    $idWeeklySchedule)
81  {
82      $statement = "
83      SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
84      FROM time_slot
85      WHERE is_deleted= :DELETED
86      AND id_educator = :ID_EDUCATOR
87      AND id_weekly_schedule = :ID_WEEKLY_SCHEDULE
88      ORDER BY code_day, time_start";
89
90      try {
91          $statement = $this->db->prepare($statement);
92          $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
93          $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
94          $statement->bindParam(':ID_WEEKLY_SCHEDULE', $idWeeklySchedule, \PDO::
            PARAM_INT);
95          $statement->execute();
96          $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
97
98          $timeSlotArray = array();
99
100         foreach ($results as $result) {
101             $timeSlot = new TimeSlot();
102             $timeSlot->id = $result["id"];
103             $timeSlot->code_day = $result["code_day"];
104             $timeSlot->time_start = $result["time_start"];
105             $timeSlot->time_end = $result["time_end"];
106             $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
107             $timeSlot->id_schedule_override = $result["id_schedule_override"];
108             $timeSlot->id_educator = $result["id_educator"];
109             array_push($timeSlotArray,$timeSlot);
110         }
111
112         return $timeSlotArray;
113     } catch (\PDOException $e) {
114         exit($e->getMessage());
115     }
116 }
117
118 /**
119  *
120  * Method to return all the time slot of the database in an array of timeslot
    objects from his schedule override identifier.
121  *
122  * @param bool $isDeleted Bool to define whether to search for existing or
    deleted timeslot
123  * @param int $idEducator The educator identifier

```

```

124     * @param int $idScheduleOverride The schedule override identifier
125     * @return TimeSlot[] A ScheduleOverride object array
126     */
127     public function findAllByIdScheduleOverride(bool $deleted,int $idEducator, int
        $idScheduleOverride)
128     {
129         $statement = "
130         SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
131         FROM time_slot
132         WHERE is_deleted= :DELETED
133         AND id_educator = :ID_EDUCATOR
134         AND id_schedule_override = :ID_SCHEDULE_OVERRIDE
135         ORDER BY time_start";
136
137         try {
138             $statement = $this->db->prepare($statement);
139             $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
140             $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
141             $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $idScheduleOverride, \
                PDO::PARAM_INT);
142             $statement->execute();
143             $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
144
145             $timeSlotArray = array();
146
147             foreach ($results as $result) {
148                 $timeSlot = new TimeSlot();
149                 $timeSlot->id = $result["id"];
150                 $timeSlot->code_day = $result["code_day"];
151                 $timeSlot->time_start = $result["time_start"];
152                 $timeSlot->time_end = $result["time_end"];
153                 $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
154                 $timeSlot->id_schedule_override = $result["id_schedule_override"];
155                 $timeSlot->id_educator = $result["id_educator"];
156                 array_push($timeSlotArray,$timeSlot);
157             }
158
159             return $timeSlotArray;
160         } catch (\PDOException $e) {
161             exit($e->getMessage());
162         }
163     }
164
165     /**
166     *
167     * Method to return a time slot from the database in a timeslot model object.
168     *
169     * @param int $id The timeslot identifier
170     * @param int $idEducator The educator identifier
171     * @return TimeSlot A TimeSlot model object containing all the result rows of
        the query
172     */
173     public function find(int $id, int $idEducator)
174     {
175         $statement = "
176         SELECT id, code_day, time_start, time_end,id_weekly_schedule,
            id_schedule_override,id_educator
177         FROM time_slot

```

```

178 WHERE id = :ID_TIMESLOT
179 AND is_deleted = 0
180 AND id_educator = :ID_EDUCATOR";
181
182 try {
183     $statement = $this->db->prepare($statement);
184     $statement->bindParam(':ID_TIMESLOT', $id, \PDO::PARAM_INT);
185     $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
186     $statement->execute();
187
188     $timeSlot = new TimeSlot();
189
190     if ($statement->rowCount()==1) {
191         $result = $statement->fetch(\PDO::FETCH_ASSOC);
192         $timeSlot->id = $result["id"];
193         $timeSlot->code_day = $result["code_day"];
194         $timeSlot->time_start = $result["time_start"];
195         $timeSlot->time_end = $result["time_end"];
196         $timeSlot->id_weekly_schedule = $result["id_weekly_schedule"];
197         $timeSlot->id_schedule_override = $result["id_schedule_override"];
198         $timeSlot->id_educator = $result["id_educator"];
199     }
200     else{
201         $timeSlot = null;
202     }
203
204     return $timeSlot;
205 } catch (\PDOException $e) {
206     exit($e->getMessage());
207 }
208 }
209
210 /**
211 *
212 * Method to insert a time slot in the database.
213 *
214 * @param TimeSlot $timeSlot The timeslot model object
215 * @return int The number of rows affected by the insert
216 */
217 public function insert(TimeSlot $timeSlot)
218 {
219     $statement = "
220     INSERT INTO time_slot (code_day, time_start, time_end, is_deleted,
221         id_weekly_schedule, id_schedule_override, id_educator)
222     VALUES (:CODE_DAY, :TIME_START, :TIME_END, 0, :ID_WEEKLY_SCHEDULE, :
223         ID_SCHEDULE_OVERRIDE, :ID_EDUCATOR)";
224
225     try {
226         $statement = $this->db->prepare($statement);
227         $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT);
228         $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::PARAM_STR);
229         $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR);
230         $statement->bindParam(':ID_WEEKLY_SCHEDULE', $timeSlot->id_weekly_schedule, \PDO::PARAM_INT);
231         $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $timeSlot->id_schedule_override, \PDO::PARAM_INT);

```

```

230         $statement->bindParam(':ID_EDUCATOR', $timeSlot->id_educator, \PDO::
           PARAM_INT);
231         $statement->execute();
232         return $statement->rowCount();
233     } catch (\PDOException $e) {
234         exit($e->getMessage());
235     }
236 }
237
238 /**
239  *
240  * Method to update a time slot in the database.
241  *
242  * @param TimeSlot $timeSlot The timeslot model object
243  * @return int The number of rows affected by the update
244  */
245 public function update(TimeSlot $timeSlot)
246 {
247     $statement = "
248     UPDATE time_slot
249     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
250     WHERE id = :ID_TIMESLOT";
251
252     try {
253         $statement = $this->db->prepare($statement);
254         $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT
        );
255         $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::
        PARAM_STR);
256         $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR
        );
257         $statement->bindParam(':ID_WEEKLY_SCHEDULE', $timeSlot->
        id_weekly_schedule, \PDO::PARAM_INT);
258         $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $timeSlot->
        id_schedule_override, \PDO::PARAM_INT);
259         $statement->bindParam(':ID_TIMESLOT', $timeSlot->id, \PDO::PARAM_INT);
260         $statement->execute();
261         return $statement->rowCount();
262     } catch (\PDOException $e) {
263         exit($e->getMessage());
264     }
265 }
266
267 /**
268  *
269  * Method to delete a time slot in the database.
270  *
271  * @param TimeSlot $timeSlot The timeslot model object
272  * @return int The number of rows affected by the delete
273  */
274 public function delete(TimeSlot $timeSlot)
275 {
276     $statement = "
277     UPDATE time_slot
278     SET is_deleted = 1
279     WHERE id = :ID_TIMESLOT";
280

```



```

281     try {
282         $statement = $this->db->prepare($statement);
283         $statement->bindParam(':ID_TIMESLOT', $timeSlot->id, \PDO::PARAM_INT);
284         $statement->execute();
285         return $statement->rowCount();
286     } catch (\PDOException $e) {
287         exit($e->getMessage());
288     }
289 }
290
291 /**
292  *
293  * Method to check if a time slot causes an overlapping problem in a weekly
294  * schedule.
295  *
296  * @param TimeSlot $timeSlot The timeslot model object
297  * @return array The associative array containing all the result rows of the
298  * query
299 */
300 public function findOverlapInWeeklySchedule(TimeSlot $timeSlot)
301 {
302     $statement = "
303     SELECT ts.code_day, ts.time_start, ts.time_end
304     FROM time_slot AS ts
305     LEFT JOIN weekly_schedule AS ws
306     ON ts.id_weekly_schedule = ws.id
307     WHERE ts.id_weekly_schedule = :ID_WEEKLY_SCHEDULE
308     AND ts.is_deleted = 0
309     AND ws.is_deleted = 0
310     AND :TIME_START < time_end
311     AND :TIME_END > time_start
312     AND code_day = :CODE_DAY
313     AND ts.id_educator = :ID_EDUCATOR";
314
315     try {
316         $statement = $this->db->prepare($statement);
317         $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT);
318         $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::PARAM_STR);
319         $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR);
320         $statement->bindParam(':ID_WEEKLY_SCHEDULE', $timeSlot->id_weekly_schedule, \PDO::PARAM_INT);
321         $statement->bindParam(':ID_EDUCATOR', $timeSlot->id_educator, \PDO::PARAM_INT);
322         $statement->execute();
323         $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
324         return $result;
325     } catch (\PDOException $e) {
326         exit($e->getMessage());
327     }
328 }
329
330 /**
331  *
332  * Method to check if a time slot causes an overlapping problem in a schedule
333  * override.
334  *

```

```

332     * @param TimeSlot $timeSlot The timeslot model object
333     * @return array The associative array containing all the result rows of the
        query
334     */
335     public function findOverlapInScheduleOverride(TimeSlot $timeSlot)
336     {
337         $statement = "
338             SELECT ts.code_day, ts.time_start, ts.time_end
339             FROM time_slot AS ts
340             LEFT JOIN schedule_override AS so
341             ON ts.id_schedule_override = so.id
342             WHERE ts.id_schedule_override = :ID_SCHEDULE_OVERRIDE
343             AND ts.is_deleted = 0
344             AND so.is_deleted = 0
345             AND :TIME_START < time_end
346             AND :TIME_END > time_start
347             AND code_day = :CODE_DAY
348             AND ts.id_educator = :ID_EDUCATOR";
349
350         try {
351             $statement = $this->db->prepare($statement);
352             $statement->bindParam(':CODE_DAY', $timeSlot->code_day, \PDO::PARAM_INT
        );
353             $statement->bindParam(':TIME_START', $timeSlot->time_start, \PDO::
        PARAM_STR);
354             $statement->bindParam(':TIME_END', $timeSlot->time_end, \PDO::PARAM_STR
        );
355             $statement->bindParam(':ID_SCHEDULE_OVERRIDE', $timeSlot->
        id_schedule_override, \PDO::PARAM_INT);
356             $statement->bindParam(':ID_EDUCATOR', $timeSlot->id_educator, \PDO::
        PARAM_INT);
357             $statement->execute();
358             $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
359             return $result;
360         } catch (\PDOException $e) {
361             exit($e->getMessage());
362         }
363     }
364
365     /**
366     *
367     * Method for generating virtual views of the application.
368     * 1. digits : Table containing the values from 0 to 9
369     * 2. numbers : Table containing numbers from 0 to 999
370     * 3. dates : Table containing dates from (today - 999 days) to (today + 1 year
        )
371     *
372     * @return void
373     */
374     private function generateViews()
375     {
376         $statement = "
377             CREATE OR REPLACE VIEW digits AS
378             SELECT 0 AS digit UNION ALL
379             SELECT 1 UNION ALL
380             SELECT 2 UNION ALL
381             SELECT 3 UNION ALL
382             SELECT 4 UNION ALL
383             SELECT 5 UNION ALL

```

```

384         SELECT 6 UNION ALL
385         SELECT 7 UNION ALL
386         SELECT 8 UNION ALL
387         SELECT 9;
388
389         CREATE OR REPLACE VIEW numbers AS
390         SELECT ones.digit + tens.digit * 10 + hundreds.digit * 100 AS number
391         FROM digits as ones, digits as tens, digits as hundreds;
392
393         CREATE OR REPLACE VIEW dates AS
394         SELECT SUBDATE(ADDDATE(CURRENT_DATE(),365), number) AS date
395         FROM numbers";
396
397         $this->db->exec($statement);
398     }
399
400     /**
401     *
402     * Method to retrieve the available valid time slots with their corresponding
403     * dates.
404     *
405     * @param int $idEducator The educator identifier
406     * @return array The associative array containing all the result rows of the
407     * query
408     */
409     public function findPlanningForEducator(int $idEducator)
410     {
411         $this->generateViews();
412
413         $statement = "
414         SELECT IF(dates.date IS NOT NULL, dates.date, so.date_schedule_override) AS
415             date,time_start,time_end
416
417         FROM time_slot AS ts
418         LEFT JOIN weekly_schedule AS ws
419         ON ws.id = ts.id_weekly_schedule
420
421         LEFT JOIN schedule_override AS so
422         ON so.id = ts.id_schedule_override
423
424         LEFT JOIN dates
425         ON DAYOFWEEK(dates.date) = ts.code_day
426         AND dates.date BETWEEN ws.date_valid_from
427         AND IF(ws.date_valid_to IS NULL, DATE_ADD(NOW(), INTERVAL 365 DAY), ws.
428             date_valid_to)
429
430         WHERE ts.is_deleted = 0 AND (so.is_deleted = 0 OR ws.is_deleted = 0)
431         AND ts.id_educator = :ID_EDUCATOR
432
433         AND (SELECT COUNT(*)
434         FROM absence AS ab
435         WHERE ab.id_educator = :ID_EDUCATOR
436         AND ab.is_deleted = 0
437         AND IF(so.date_schedule_override IS NULL,dates.date,so.
438             date_schedule_override) BETWEEN ab.date_absence_from AND ab.
439             date_absence_to LIMIT 1) = 0
440
441         AND (SELECT COUNT(*)
442         FROM appointment AS ap

```

```

437 WHERE ap.user_id_educator = :ID_EDUCATOR
438 AND datetime_deletion IS NULL
439 AND user_id_deletion IS NULL
440 AND DATE(ap.datetime_appointment) = IF(so.date_schedule_override IS NULL,
    dates.date,so.date_schedule_override)
441 AND TIME(ap.datetime_appointment) = ts.time_start
442 AND TIME(ADDTIME(ap.datetime_appointment,SEC_TO_TIME(3600* ap.
    duration_in_hour))) = ts.time_end LIMIT 1) = 0
443
444 AND IF(so.date_schedule_override IS NULL,dates.date > NOW() ,so.
    date_schedule_override >NOW())
445
446 AND (SELECT COUNT(*)
447     FROM schedule_override
448     WHERE schedule_override.date_schedule_override = dates.date
449     AND schedule_override.is_deleted = 0
450     AND schedule_override.id_educator = :ID_EDUCATOR ) = 0
451
452 ORDER BY date,time_start";
453
454 try {
455     $statement = $this->db->prepare($statement);
456     $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
457     $statement->execute();
458     $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
459     return $result;
460 } catch (\PDOException $e) {
461     exit($e->getMessage());
462 }
463 }
464
465 /**
466  *
467  * Method to check if appointment data is valid in an educator's planning.
468  *
469  * @param string $date The date of the appointment
470  * @param string $time_start The start time of the appointment
471  * @param string $time_end The end time of the appointment
472  * @param int $idEducator The educator identifier
473  * @return array The associative array containing all the result rows of the
    query
474  */
475 public function findAppointmentSlotsForEducator(string $date,string $time_start
    , string $time_end,int $idEducator)
476 {
477     $this->generateViews();
478
479     $statement = "
480     SELECT IF(dates.date IS NOT NULL, dates.date, so.date_schedule_override) AS
        date,time_start,time_end
481
482     FROM time_slot AS ts
483     LEFT JOIN weekly_schedule AS ws
484     ON ws.id = ts.id_weekly_schedule
485
486     LEFT JOIN schedule_override AS so
487     ON so.id = ts.id_schedule_override
488
489     LEFT JOIN dates

```

```

490     ON DAYOFWEEK(dates.date) = ts.code_day
491     AND dates.date BETWEEN ws.date_valid_from
492     AND IF(ws.date_valid_to IS NULL, DATE_ADD(NOW(), INTERVAL 365 DAY), ws.
        date_valid_to)
493
494     WHERE ts.is_deleted = 0 AND (so.is_deleted = 0 OR ws.is_deleted = 0)
495     AND ts.id_educator = :ID_EDUCATOR
496
497
498     AND (SELECT COUNT(*)
499     FROM absence AS ab
500     WHERE ab.id_educator = :ID_EDUCATOR
501     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
502
503     AND (SELECT COUNT(*)
504     FROM appointment AS ap
505     WHERE ap.user_id_educator = :ID_EDUCATOR
506     AND ap.user_id_deletion IS NULL
507     AND ap.datetime_deletion IS NULL
508     AND DATE(ap.datetime_appointment) = IF(so.date_schedule_override IS NULL,
        dates.date, so.date_schedule_override)
509     AND TIME(ap.datetime_appointment) = ts.time_start
510     AND TIME(ADDTIME(ap.datetime_appointment, SEC_TO_TIME(3600* ap.
        duration_in_hour))) = ts.time_end LIMIT 1) = 0
511
512     AND IF(so.date_schedule_override IS NULL, dates.date > NOW(), so.
        date_schedule_override > NOW())
513
514     AND (SELECT COUNT(*)
515     FROM schedule_override
516     WHERE schedule_override.date_schedule_override = dates.date
517     AND schedule_override.is_deleted = 0
518     AND schedule_override.id_educator = :ID_EDUCATOR ) = 0
519
520     HAVING DATE = :DATE
521     AND time_start = :TIME_START
522     AND time_end = :TIME_END
523
524     ORDER BY date, time_start";
525
526     try {
527         $statement = $this->db->prepare($statement);
528         $statement->bindParam(':DATE', $date, \PDO::PARAM_STR);
529         $statement->bindParam(':TIME_START', $time_start, \PDO::PARAM_STR);
530         $statement->bindParam(':TIME_END', $time_end, \PDO::PARAM_STR);
531         $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
532         $statement->execute();
533         $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
534         return $result;
535     } catch (\PDOException $e) {
536         exit($e->getMessage());
537     }
538 }
539 }

```

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

```

1 <?php
2 /**
3  * DAOUser.php
4  *
5  * Data access object of the user table.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\DataAccessObject;
10
11 use App\Models\User;
12
13 class DAOUser {
14
15     private \PDO $db;
16
17     /**
18      *
19      * Constructor of the DAOUser object.
20      *
21      * @param PDO $db The database connection
22      */
23     public function __construct(\PDO $db)
24     {
25         $this->db = $db;
26     }
27
28     /**
29      *
30      * Method to return all users with the selected role from the database in an
31      * array of user objects.
32      *
33      * @return User[] A User object array
34      */
35     public function findAll(int $code_role)
36     {
37         $statement = "
38         SELECT id, email, firstname, lastname, phonenumber, address
39         FROM user
40         WHERE code_role = :CODE_ROLE";
41
42         try {
43             $statement = $this->db->prepare($statement);
44             $statement->bindParam(':CODE_ROLE', $code_role, \PDO::PARAM_INT);
45             $statement->execute();
46             $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
47
48             $userArray = array();
49
50             foreach ($results as $result) {
51                 $user = new User();
52                 $user->id = $result["id"];
53                 $user->email = $result["email"];
54                 $user->firstname = $result["firstname"];
55                 $user->lastname = $result["lastname"];
56                 $user->phonenumber = $result["phonenumber"];
57                 $user->address = $result["address"];
58                 array_push($userArray, $user);
59             }
60         }
61     }
62 }

```

```

59         return $userArray;
60     } catch (\PDOException $e) {
61         exit($e->getMessage());
62     }
63 }
64
65 /**
66  *
67  * Method to return all users with the administrator role from the database in
68  * an array of user objects.
69  *
70  * @return User[] A User object array
71  */
72 public function findAllEducators()
73 {
74     $statement = "
75     SELECT id, firstname, lastname
76     FROM user
77     WHERE code_role = 2";
78
79     try {
80         $statement = $this->db->prepare($statement);
81         $statement->bindParam(':CODE_ROLE', $code_role, \PDO::PARAM_INT);
82         $statement->execute();
83         $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
84
85         $userArray = array();
86
87         foreach ($results as $result) {
88             $user = new User();
89             $user->id = $result["id"];
90             $user->firstname = $result["firstname"];
91             $user->lastname = $result["lastname"];
92             array_push($userArray, $user);
93         }
94         return $userArray;
95     } catch (\PDOException $e) {
96         exit($e->getMessage());
97     }
98 }
99
100 /**
101  *
102  * Method to return a user from the database in a user model object.
103  *
104  * @param int $id The user identifier
105  * @return User A User model object containing all the result rows of the query
106  */
107 public function find(int $id)
108 {
109     $statement = "
110     SELECT id, email, firstname, lastname, phonenumber, address, api_token,
111     code_role, password_hash
112     FROM user
113     WHERE id = :ID_USER";
114
115     try {
116         $statement = $this->db->prepare($statement);
117         $statement->bindParam(':ID_USER', $id, \PDO::PARAM_INT);

```

```

116         $statement->execute();
117
118         $user = new User();
119
120         if ($statement->rowCount()==1) {
121             $result = $statement->fetch(\PDO::FETCH_ASSOC);
122             $user->id = $result["id"];
123             $user->email = $result["email"];
124             $user->firstname = $result["firstname"];
125             $user->lastname = $result["lastname"];
126             $user->phonenummer = $result["phonenummer"];
127             $user->address = $result["address"];
128             $user->api_token = $result["api_token"];
129             $user->code_role = $result["code_role"];
130             $user->password_hash = $result["password_hash"];
131         }
132         else{
133             $user = null;
134         }
135
136         return $user;
137     } catch (\PDOException $e) {
138         exit($e->getMessage());
139     }
140 }
141
142 /**
143  *
144  * Method to insert a user in the database.
145  *
146  * @param User $user The user model object
147  * @return int The number of rows affected by the insert
148  */
149 public function insert(User $user)
150 {
151     $statement = "
152     INSERT INTO user (email, firstname, lastname, phonenummer, address,
153     api_token, code_role, password_hash)
154     VALUES (:EMAIL, :FIRSTNAME, :LASTNAME, :PHONENUMBER, :ADDRESS, :API_TOKEN, :
155     CODE_ROLE, :PASSWORD_HASH)";
156
157     try {
158         $statement = $this->db->prepare($statement);
159         $statement->bindParam(':EMAIL', $user->email, \PDO::PARAM_STR);
160         $statement->bindParam(':FIRSTNAME', $user->firstname, \PDO::PARAM_STR);
161         $statement->bindParam(':LASTNAME', $user->lastname, \PDO::PARAM_STR);
162         $statement->bindParam(':PHONENUMBER', $user->phonenummer, \PDO::
163         PARAM_STR);
164         $statement->bindParam(':ADDRESS', $user->address, \PDO::PARAM_STR);
165         $statement->bindParam(':API_TOKEN', $user->api_token, \PDO::PARAM_STR);
166         $statement->bindParam(':CODE_ROLE', $user->code_role, \PDO::PARAM_INT);
167         $statement->bindParam(':PASSWORD_HASH', $user->password_hash, \PDO::
168         PARAM_STR);
169         $statement->execute();
170         return $statement->rowCount();
171     } catch (\PDOException $e) {
172         exit($e->getMessage());
173     }
174 }

```



```

171
172 /**
173  *
174  * Method to update a user in the database.
175  *
176  * @param User $user The user model object
177  * @return int The number of rows affected by the update
178  */
179 public function update(User $user)
180 {
181     $statement = "
182     UPDATE user
183     SET email = :EMAIL,
184     firstname = :FIRSTNAME,
185     lastname = :LASTNAME,
186     phonenumber = :PHONENUMBER,
187     address = :ADDRESS,
188     api_token = :API_TOKEN,
189     code_role = :CODE_ROLE,
190     password_hash = :PASSWORD_HASH
191     WHERE id = :ID_USER";
192
193     try {
194         $statement = $this->db->prepare($statement);
195         $statement->bindParam(':EMAIL', $user->email, \PDO::PARAM_STR);
196         $statement->bindParam(':FIRSTNAME', $user->firstname, \PDO::PARAM_STR);
197         $statement->bindParam(':LASTNAME', $user->lastname, \PDO::PARAM_STR);
198         $statement->bindParam(':PHONENUMBER', $user->phonenumber, \PDO::
199             PARAM_STR);
200         $statement->bindParam(':ADDRESS', $user->address, \PDO::PARAM_STR);
201         $statement->bindParam(':API_TOKEN', $user->api_token, \PDO::PARAM_STR);
202         $statement->bindParam(':CODE_ROLE', $user->code_role, \PDO::PARAM_STR);
203         $statement->bindParam(':PASSWORD_HASH', $user->password_hash, \PDO::
204             PARAM_STR);
205         $statement->bindParam(':ID_USER', $user->id, \PDO::PARAM_INT);
206         $statement->execute();
207         return $statement->rowCount();
208     } catch (\PDOException $e) {
209         exit($e->getMessage());
210     }
211 }
212
213 /**
214  *
215  * Method to delete a user in the database.
216  *
217  * @param User $user The user model object
218  * @return int The number of rows affected by the delete
219  */
220 public function delete(User $user)
221 {
222     $statement = "
223     DELETE FROM user
224     WHERE id = :ID_USER";
225
226     try {
227         $statement = $this->db->prepare($statement);
228         $statement->bindParam(':ID_USER', $user->id, \PDO::PARAM_INT);
229         $statement->execute();

```

```

228         return $statement->rowCount();
229     } catch (\PDOException $e) {
230         exit($e->getMessage());
231     }
232 }
233
234 /**
235  *
236  * Method to return a user from the database user from his api token.
237  *
238  * @param string $api_token The user api token
239  * @return User A User model object containing all the result rows of the query
240  */
241 public function findByApiToken(string $api_token)
242 {
243     $statement = "
244     SELECT id, email, firstname, lastname, phonenumber, address, api_token,
245           code_role,password_hash
246     FROM user
247     WHERE api_token = :API_TOKEN
248     LIMIT 1";
249
250     try {
251         $statement = $this->db->prepare($statement);
252         $statement->bindParam(':API_TOKEN', $api_token, \PDO::PARAM_STR);
253         $statement->execute();
254
255         $user = new User();
256
257         if ($statement->rowCount()==1) {
258             $result = $statement->fetch(\PDO::FETCH_ASSOC);
259             $user->id = $result["id"];
260             $user->email = $result["email"];
261             $user->firstname = $result["firstname"];
262             $user->lastname = $result["lastname"];
263             $user->phonenumber = $result["phonenumber"];
264             $user->address = $result["address"];
265             $user->api_token = $result["api_token"];
266             $user->code_role = $result["code_role"];
267         }
268         else{
269             $user = null;
270         }
271
272         return $user;
273     } catch (\PDOException $e) {
274         exit($e->getMessage());
275     }
276 }
277
278 /**
279  *
280  * Method to return a user from the database user from his email.
281  *
282  * @param string $email The user email
283  * @return User A User model object containing all the result rows of the query
284  */
285 public function findUserByEmail(string $email)
286 {

```

```

286     $statement = "
287     SELECT id, email, firstname, lastname, phonenumber, address, api_token,
        code_role, password_hash
288     FROM user
289     WHERE email = :EMAIL
290     LIMIT 1";
291
292     try {
293         $statement = $this->db->prepare($statement);
294         $statement->bindParam(':EMAIL', $email, \PDO::PARAM_STR);
295         $statement->execute();
296
297         $user = new User();
298
299         if ($statement->rowCount()==1) {
300             $result = $statement->fetch(\PDO::FETCH_ASSOC);
301             $user->id = $result["id"];
302             $user->email = $result["email"];
303             $user->firstname = $result["firstname"];
304             $user->lastname = $result["lastname"];
305             $user->phonenumber = $result["phonenumber"];
306             $user->address = $result["address"];
307             $user->api_token = $result["api_token"];
308             $user->code_role = $result["code_role"];
309             $user->password_hash = $result["password_hash"];
310         }
311         else {
312             $user = null;
313         }
314
315         return $user;
316     } catch (\PDOException $e) {
317         exit($e->getMessage());
318     }
319 }
320 }

```

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

```

1  <?php
2  /**
3   * DAOWeeklySchedule.php
4   *
5   * Data access object of the weekly_schedule table.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\DataAccessObject;
10
11  use App\Models\WeeklySchedule;
12
13  class DAOWeeklySchedule {
14
15      private $db = null;
16
17      /**
18       *
19       * Constructor of the DAOWeeklySchedule object.
20       *

```

```

21     * @param PDO $db The database connection
22     */
23     public function __construct(\PDO $db)
24     {
25         $this->db = $db;
26     }
27
28     /**
29     *
30     * Method to return all the weekly schedule of the database in an array of
31     * weeklyschedules objects.
32     *
33     * @param bool $isDeleted Bool to define whether to search for existing or
34     * deleted weeklyschedules
35     * @param int $idEducator The educator identifier
36     * @return WeeklySchedule[] A WeeklySchedule object array
37     */
38     public function findAll(bool $deleted, int $idEducator)
39     {
40         $statement = "
41         SELECT id, date_valid_from, date_valid_to, id_educator
42         FROM weekly_schedule
43         WHERE is_deleted= :DELETED
44         AND id_educator = :ID_EDUCATOR
45         ORDER BY date_valid_from";
46
47         try {
48             $statement = $this->db->prepare($statement);
49             $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
50             $statement->bindParam(':DELETED', $deleted, \PDO::PARAM_BOOL);
51             $statement->execute();
52             $results = $statement->fetchAll(\PDO::FETCH_ASSOC);
53
54             $weeklyScheduleArray = array();
55
56             foreach ($results as $result) {
57                 $weeklySchedule = new WeeklySchedule();
58                 $weeklySchedule->id = $result["id"];
59                 $weeklySchedule->date_valid_from = $result["date_valid_from"];
60                 $weeklySchedule->date_valid_to = $result["date_valid_to"];
61                 $weeklySchedule->id_educator = $result["id_educator"];
62                 array_push($weeklyScheduleArray, $weeklySchedule);
63             }
64
65             return $weeklyScheduleArray;
66         } catch (\PDOException $e) {
67             exit($e->getMessage());
68         }
69     }
70
71     /**
72     *
73     * Method to return an weekly schedule from the database in a weeklyschedules
74     * model object.
75     *
76     * @param int $id The weeklyschedules identifier
77     * @param int $idEducator The educator identifier
78     * @return WeeklySchedule A WeeklySchedule model object containing all the
79     * result rows of the query

```

```

76     */
77     public function find(int $id, int $idEducator)
78     {
79         $statement = "
80         SELECT id, date_valid_from, date_valid_to, id_educator
81         FROM weekly_schedule
82         WHERE id = :ID_WEEKLY_SCHEDULE
83         AND is_deleted = 0
84         AND id_educator = :ID_EDUCATOR";
85
86         try {
87             $statement = $this->db->prepare($statement);
88             $statement->bindParam(':ID_WEEKLY_SCHEDULE', $id, \PDO::PARAM_INT);
89             $statement->bindParam(':ID_EDUCATOR', $idEducator, \PDO::PARAM_INT);
90             $statement->execute();
91
92             $weeklySchedule = new WeeklySchedule();
93
94             if ($statement->rowCount() == 1) {
95                 $result = $statement->fetch(\PDO::FETCH_ASSOC);
96                 $weeklySchedule->id = $result["id"];
97                 $weeklySchedule->date_valid_from = $result["date_valid_from"];
98                 $weeklySchedule->date_valid_to = $result["date_valid_to"];
99                 $weeklySchedule->id_educator = $result["id_educator"];
100             }
101             else{
102                 $weeklySchedule = null;
103             }
104
105             return $weeklySchedule;
106         } catch (\PDOException $e) {
107             exit($e->getMessage());
108         }
109     }
110
111     /**
112     *
113     * Method to insert a weekly schedule in the database.
114     *
115     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
116     * @return int The number of rows affected by the insert
117     */
118     public function insert(WeeklySchedule $weeklySchedule)
119     {
120         $statement = "
121         INSERT INTO weekly_schedule (date_valid_from, date_valid_to, id_educator,
122         is_deleted)
123         VALUES (STR_TO_DATE(:DATE_VALID_FROM, \"%Y-%m-%d\"), STR_TO_DATE(:
124         DATE_VALID_TO, \"%Y-%m-%d\"), :ID_EDUCATOR, 0)";
125
126         try {
127             $statement = $this->db->prepare($statement);
128             $statement->bindParam(':DATE_VALID_FROM', $weeklySchedule->
129             date_valid_from, \PDO::PARAM_STR);
130             $statement->bindParam(':DATE_VALID_TO', $weeklySchedule->date_valid_to,
131             \PDO::PARAM_STR);
132             $statement->bindParam(':ID_EDUCATOR', $weeklySchedule->id_educator, \
133             PDO::PARAM_INT);
134             $statement->execute();

```

```

130         return $statement->rowCount();
131     } catch (\PDOException $e) {
132         exit($e->getMessage());
133     }
134 }
135
136 /**
137  *
138  * Method to update a weekly schedule in the database.
139  *
140  * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
141  * @return int The number of rows affected by the update
142  */
143 public function update(WeeklySchedule $weeklySchedule)
144 {
145     $statement = "
146     UPDATE weekly_schedule
147     SET date_valid_from = STR_TO_DATE(:DATE_VALID_FROM, \"%Y-%m-%d\"),
148     date_valid_to = STR_TO_DATE(:DATE_VALID_TO, \"%Y-%m-%d\")
149     WHERE id = :ID_WEEKLY_SCHEDULE";
150
151     try {
152         $statement = $this->db->prepare($statement);
153         $statement->bindParam(':DATE_VALID_FROM', $weeklySchedule->
154             date_valid_from, \PDO::PARAM_STR);
155         $statement->bindParam(':DATE_VALID_TO', $weeklySchedule->date_valid_to,
156             \PDO::PARAM_STR);
157         $statement->bindParam(':ID_WEEKLY_SCHEDULE', $weeklySchedule->id, \PDO
158             ::PARAM_INT);
159         $statement->execute();
160         return $statement->rowCount();
161     } catch (\PDOException $e) {
162         exit($e->getMessage());
163     }
164 }
165
166 /**
167  *
168  * Method to delete a weekly schedule in the database.
169  *
170  * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
171  * @return int The number of rows affected by the delete
172  */
173 public function delete(WeeklySchedule $weeklySchedule)
174 {
175     $statement = "
176     UPDATE weekly_schedule
177     SET is_deleted = 1
178     WHERE id = :ID_WEEKLY_SCHEDULE";
179
180     try {
181         $statement = $this->db->prepare($statement);
182         $statement->bindParam(':ID_WEEKLY_SCHEDULE', $weeklySchedule->id, \PDO
183             ::PARAM_INT);
184         $statement->execute();
185         return $statement->rowCount();
186     } catch (\PDOException $e) {
187         exit($e->getMessage());
188     }
189 }

```

```

185     }
186
187     /**
188     *
189     * Method to check if a start date and an end date of a weekly schedule do not
190     * cause an overlapping problem.
191     *
192     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
193     * @return array The associative array containing all the result rows of the
194     * query
195     */
196     public function findOverlap(WeeklySchedule $weeklySchedule)
197     {
198         $statement = "
199         SELECT *
200         FROM weekly_schedule
201         WHERE (STR_TO_DATE(:DATE_VALID_FROM, \"%Y-%m-%d\") < date_valid_to OR
202               date_valid_to IS NULL)
203         AND (STR_TO_DATE(:DATE_VALID_TO, \"%Y-%m-%d\") > date_valid_from
204         OR (STR_TO_DATE(:DATE_VALID_TO, \"%Y-%m-%d\") IS NULL AND (STR_TO_DATE(:
205             DATE_VALID_FROM, \"%Y-%m-%d\") < date_valid_to)))
206         AND id_educator = :ID_EDUCATOR
207         AND is_deleted = 0";
208
209         if (!isset($input['date_valid_to'])) {
210             $input['date_valid_to'] = null;
211         }
212
213         try {
214             $statement = $this->db->prepare($statement);
215             $statement->bindParam(':DATE_VALID_FROM', $weeklySchedule->
216                 date_valid_from, \PDO::PARAM_STR);
217             $statement->bindParam(':DATE_VALID_TO', $weeklySchedule->date_valid_to,
218                 \PDO::PARAM_STR);
219             $statement->bindParam(':ID_EDUCATOR', $weeklySchedule->id_educator, \
220                 PDO::PARAM_INT);
221             $statement->execute();
222             $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
223             return $result;
224         } catch (\PDOException $e) {
225             exit($e->getMessage());
226         }
227     }
228
229     /**
230     *
231     * Method to check if the database contains a weekly schedule not deleted
232     * without end date.
233     *
234     * @param WeeklySchedule $weeklySchedule The weeklyschedule model object
235     * @return array The associative array containing all the result rows of the
236     * query
237     */
238     public function findActifPermanentSchedule(WeeklySchedule $weeklySchedule)
239     {
240         $statement = "
241         SELECT *
242         FROM weekly_schedule
243         WHERE date_valid_to IS NULL

```

```

235         AND is_deleted = 0
236         AND id != :ID_HIMSELF
237         AND id_educator = :ID_EDUCATOR";
238
239     try {
240         $statement = $this->db->prepare($statement);
241         $statement->bindParam(':ID_EDUCATOR', $weeklySchedule->id_educator, \
                PDO::PARAM_INT);
242         $statement->bindParam(':ID_HIMSELF', $weeklySchedule->id, \PDO::
                PARAM_INT);
243         $statement->execute();
244         $result = $statement->fetchAll(\PDO::FETCH_ASSOC);
245         return $result;
246     } catch (\PDOException $e) {
247         exit($e->getMessage());
248     }
249 }
250 }

```

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

```

1  <?php
2  /**
3   * Absence.php
4   *
5   * Absence model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\Models;
10
11  class Absence {
12
13      public ?int $id;
14      public ?string $date_absence_from;
15      public ?string $date_absence_to;
16      public ?string $description;
17      public ?int $id_educator;
18
19      /**
20       *
21       * Constructor of the Absence model object.
22       *
23       * @param int $id The Absence identifier
24       * @param string $date_absence_from The start date of the absences
25       * @param string $date_absence_to The end date of the absences
26       * @param string $description The description of the absences
27       * @param int $id_educator The educator identifier
28       */
29      public function __construct(int $id = null, string $date_absence_from = null,
                string $date_absence_to = null, string $description = null, int $id_educator
                = null)
30      {
31          $this->id = $id;
32          $this->date_absence_from = $date_absence_from;
33          $this->date_absence_to = $date_absence_to;
34          $this->description = $description;
35          $this->id_educator = $id_educator;
36      }

```



37 }

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

```
1 <?php
2 /**
3  * Appointment.php
4  *
5  * Appointment model.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\Models;
10
11 class Appointment {
12
13     public ?int $id;
14     public ?string $datetime_appointment;
15     public ?int $duration_in_hour;
16     public ?string $note_text;
17     public ?string $note_graphical_serial_id;
18     public ?string $summary;
19     public ?int $user_id_customer;
20     public ?int $user_id_educator;
21
22     /**
23      *
24      * Constructor of the Appointment model object.
25      *
26      * @param int $id The Appointment identifier
27      * @param string $datetime_appointment The date of the appointment
28      * @param int $duration_in_hour The duration of the appointment in hours
29      * @param string $note_text The text note of the appointment
30      * @param string $note_graphical_serial_id The serial id of the graphical note
31      * @param string $summary The educator identifier
32      * @param int $user_id_customer The customer identifier
33      * @param int $user_id_educator The educator identifier
34      */
35     public function __construct(int $id = null, string $datetime_appointment = null
36         , int $duration_in_hour = null, string $note_text = null, string
37         $note_graphical_serial_id = null, string $summary = null, int
38         $user_id_customer = null, int $user_id_educator = null)
39     {
40         $this->id = $id;
41         $this->datetime_appointment = $datetime_appointment;
42         $this->duration_in_hour = $duration_in_hour;
43         $this->note_text = $note_text;
44         $this->note_graphical_serial_id = $note_graphical_serial_id;
45         $this->summary = $summary;
46         $this->user_id_customer = $user_id_customer;
47         $this->user_id_educator = $user_id_educator;
48     }
49 }
```

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

```
1 <?php
2 /**
3  * Document.php
```

```

4  *
5  * Document model.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9  namespace App\Models;
10
11 class Document {
12
13     public ?int $id;
14     public ?string $document_serial_id;
15     public ?string $type;
16     public ?int $user_id;
17     public ?int $package_number;
18     public ?string $signature_base64;
19
20     /**
21      *
22      * Constructor of the Document model object.
23      *
24      * @param int $id The document identifier
25      * @param string $document_serial_id The serial id of the document
26      * @param string $type The type of the document
27      * @param int $user_id The identifier of the owner of the document
28      * @param int $package_number The package number (Number from 1 to 5)
29      * @param string $signature_base64 The image of the signature in base64
30      */
31     public function __construct(int $id = null, string $document_serial_id = null,
32         string $type = null, int $user_id = null, int $package_number = null, string
33         $signature_base64 = null)
34     {
35         $this->id = $id;
36         $this->document_serial_id = $document_serial_id;
37         $this->type = $type;
38         $this->user_id = $user_id;
39         $this->package_number = $package_number;
40         $this->signature_base64 = $signature_base64;
41     }
42 }

```

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

```

1  <?php
2  /**
3   * Dog.php
4   *
5   * Dog model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\Models;
10
11 class Dog {
12
13     public ?int $id;
14     public ?string $name;
15     public ?string $breed;
16     public ?string $sex;
17     public ?string $picture_serial_id;

```

```

18     public ?string $chip_id;
19     public ?int $user_id;
20
21     /**
22      *
23      * Constructor of the Dog model object.
24      *
25      * @param int $id The dog identifier
26      * @param string $name The name of the dog
27      * @param string $breed The breed of the dog
28      * @param string $sex The sex of the dog
29      * @param string $picture_serial_id The picture serial id of the dog
30      * @param string $chip_id The chip id of the dog
31      * @param int $user_id The user id of the dog's owner
32      */
33     public function __construct(int $id = null, string $name = null, string $breed
        = null,
34         string $sex = null, string $picture_serial_id = null, string $chip_id = null,
            int $user_id = null)
35     {
36         $this->id = $id;
37         $this->name = $name;
38         $this->breed = $breed;
39         $this->sex = $sex;
40         $this->picture_serial_id = $picture_serial_id;
41         $this->chip_id = $chip_id;
42         $this->user_id = $user_id;
43     }
44 }

```

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

```

1 <?php
2 /**
3  * ScheduleOverride.php
4  *
5  * ScheduleOverride model.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 namespace App\Models;
10
11 class ScheduleOverride {
12
13     public ?int $id;
14     public ?string $date_schedule_override;
15     public ?int $id_educator;
16
17     /**
18      *
19      * Constructor of the ScheduleOverride model object.
20      *
21      * @param int $id The ScheduleOverride identifier
22      * @param string $date_valid_from The date of the schedule override
23      * @param int $id_educator The educator identifier
24      */
25     public function __construct(int $id = null, string $date_schedule_override =
        null ,int $id_educator = null)
26     {

```

```

27         $this->id = $id;
28         $this->date_schedule_override = $date_schedule_override;
29         $this->id_educator = $id_educator;
30     }
31 }

```

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

```

1  <?php
2  /**
3   * TimeSlot.php
4   *
5   * TimeSlot model.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\Models;
10
11  class TimeSlot {
12
13      public ?int $id;
14      public ?int $code_day;
15      public ?string $time_start;
16      public ?string $time_end;
17      public ?int $id_weekly_schedule;
18      public ?int $id_schedule_override;
19      public ?int $id_educator;
20
21      /**
22       *
23       * Constructor of the TimeSlot model object.
24       *
25       * @param int $id The TimeSlot identifier
26       * @param int $code_day The day code of a week (1 = Sunday, 2 = Monday, 3 =
27         Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, 7 = Saturday)
28       * @param string $time_start The start time of the time slot
29       * @param string $time_end The end time of the time slot
30       * @param int $id_weekly_schedule The weekly schedule identifier
31       * @param int $id_schedule_override The schedule override identifier
32       * @param int $id_educator The educator identifier
33       */
34      public function __construct(int $id = null, int $code_day = null, string
35         $time_start = null, string $time_end = null, int $id_weekly_schedule = null
36         , int $id_schedule_override = null, int $id_educator = null)
37      {
38          $this->id = $id;
39          $this->code_day = $code_day;
40          $this->time_start = $time_start;
41          $this->time_end = $time_end;
42          $this->id_weekly_schedule = $id_weekly_schedule;
43          $this->id_schedule_override = $id_schedule_override;
44          $this->id_educator = $id_educator;
45      }
46  }

```

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

```

1  <?php
2  /**

```

```

3  * User.php
4  *
5  * User model.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9  namespace App\Models;
10
11 class User {
12
13     public ?int $id;
14     public ?string $email;
15     public ?string $firstname;
16     public ?string $lastname;
17     public ?string $phonenumber;
18     public ?string $address;
19     public ?string $api_token;
20     public ?int $code_role;
21     public ?string $password_hash;
22
23     /**
24      *
25      * Constructor of the User model object.
26      *
27      * @param int $id The user identifier
28      * @param string $email The email of the user
29      * @param string $firstname the first name of the user
30      * @param string $lastname the last name of the user
31      * @param string $phonenumber The phone number of the user
32      * @param string $address The address of the user
33      * @param string $api_token The api_token of the user
34      * @param int $code_role The code_role of the user
35      * @param string $password_hash The password_hash of the user
36      */
37     public function __construct(int $id = null, string $email = null, string
38         $firstname = null,
39         string $lastname = null, string $phonenumber = null, string $address = null,
40         string $api_token = null, int $code_role = null, string $password_hash =
41         null)
42     {
43         $this->id = $id;
44         $this->email = $email;
45         $this->firstname = $firstname;
46         $this->lastname = $lastname;
47         $this->phonenumber = $phonenumber;
48         $this->address = $address;
49         $this->api_token = $api_token;
50         $this->code_role = $code_role;
51         $this->password_hash = $password_hash;
52     }
53 }

```

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

```

1  <?php
2  /**
3   * WeeklySchedule.php
4   *
5   * WeeklySchedule model.

```

```

6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9  namespace App\Models;
10
11 class WeeklySchedule {
12
13     public ?int $id;
14     public ?string $date_valid_from;
15     public ?string $date_valid_to;
16     public ?int $id_educator;
17
18     /**
19      *
20      * Constructor of the WeeklySchedule model object.
21      *
22      * @param int $id The WeeklySchedule identifier
23      * @param string $date_valid_from The start date of the weekly schedule
24      * @param string $date_valid_to The end date of the weekly schedule
25      * @param int $id_educator The educator identifier
26      */
27     public function __construct(int $id = null, string $date_valid_from = null,
28         string $date_valid_to = null, int $id_educator = null)
29     {
30         $this->id = $id;
31         $this->date_valid_from = $date_valid_from;
32         $this->date_valid_to = $date_valid_to;
33         $this->id_educator = $id_educator;
34     }
35 }

```

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

```

1  <?php
2  /**
3   * Constants.php
4   *
5   * Class allowing the use of the constants of the application.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9  namespace App\System;
10
11 class Constants
12 {
13     const USER_CODE_ROLE = 1;
14     const ADMIN_CODE_ROLE = 2;
15     const DOCUMENT_TYPE_CONDITIONS_OF_REGISTRATION = "conditions_inscription";
16     const DOCUMENT_TYPE_POSTER = "poster";
17     const IMAGE_TYPE_JPEG = "image/jpeg";
18     const IMAGE_TYPE_PNG = "image/png";
19     const TYPE_DOCUMENT_PDF = "application/pdf";
20 }

```

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

```

1  <?php
2  /**
3   * DatabaseConnector.php

```

```

4  *
5  * Class allowing the connection to the database with the environment variables of
   the .env file.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9  namespace App\System;
10
11 class DatabaseConnector {
12
13     private \PDO $dbConnection;
14
15     /**
16      *
17      * Constructor of the User object.
18      *
19      */
20     public function __construct()
21     {
22         $host = getenv('DB_HOST');
23         $port = getenv('DB_PORT');
24         $db = getenv('DB_DATABASE');
25         $user = getenv('DB_USERNAME');
26         $password = getenv('DB_PASSWORD');
27
28         try {
29             $this->dbConnection = new \PDO("mysql:host=$host;port=$port;charset=
               utf8mb4;dbname=$db", $user, $password);
30         } catch (\PDOException $e) {
31             exit($e->getMessage());
32         }
33     }
34
35     /**
36      *
37      * Method for returning the connection.
38      *
39      * @return PDO The database connection
40      */
41     public function getConnection()
42     {
43         return $this->dbConnection;
44     }
45 }

```

Listing 29 – ./Sources/bootstrap.php

```

1  <?php
2  /**
3   * bootstrap.php
4   *
5   * Bootable file of the API.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 require 'vendor/autoload.php';
11 use Dotenv\Dotenv;
12 use App\System\DatabaseConnector;

```

```

13
14 // Loads the environment variables from the .env file
15 $dotenv = new DotEnv(__DIR__);
16 $dotenv->load();
17
18 $dbConnection = (new DatabaseConnector())->getConnection();

```

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

```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to process absence
    requests.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\AbsenceController;
11 use App\Models\Absence;
12
13 require "../../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,PATCH,DELETE');
25     return;
26 }
27
28 $controller = new AbsenceController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 parse_str(file_get_contents('php://input'), $input);
35
36 $absence = new Absence();
37 $absence->id = $id ?? null;
38 $absence->date_absence_from = $input["date_absence_from"] ?? null;
39 $absence->date_absence_to = $input["date_absence_to"] ?? null;
40 $absence->description = $input["description"] ?? null;
41
42 switch ($requestMethod) {
43     case 'GET':
44         if (empty($id) || !is_numeric($id)) {
45             $response = $controller->getAllAbsences();
46         }
47         else{
48             $response = $controller->getAbsence($id);

```



```

49     }
50     break;
51
52     case 'POST':
53         $response = $controller->createAbsence($absence);
54         break;
55
56     case 'PATCH':
57         if (empty($id) || !is_numeric($id)) {
58             header("HTTP/1.1 404 Not Found");
59             exit();
60         }
61         $response = $controller->updateAbsence($absence);
62         break;
63
64     case 'DELETE':
65         if (empty($id) || !is_numeric($id)) {
66             header("HTTP/1.1 404 Not Found");
67             exit();
68         }
69         $response = $controller->deleteAbsence($id);
70         break;
71
72     default:
73         header("HTTP/1.1 404 Not Found");
74         exit();
75         break;
76 }
77
78 header($response['status_code_header']);
79 if ($response['body']) {
80     echo $response['body'];
81 }

```

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

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process download note
6   * graphical.
7   *
8   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
9   */
10 use App\Controllers\AppointmentController;
11
12 require "../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: GET");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
19     Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];

```

```

22 if (strcmp("OPTIONS", $requestMethod) == 0) {
23     header('Allow: GET,POST,PATCH,DELETE');
24     return;
25 }
26
27 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
28 $pathFragments = explode('/', $path);
29 $serial_id = end($pathFragments);
30
31 $controller = new AppointmentController($dbConnection);
32
33 switch ($requestMethod) {
34     case 'GET':
35         if (empty($serial_id)) {
36             header("HTTP/1.1 404 Not Found");
37             exit();
38         }
39         $response = $controller->downloadNoteGraphical($serial_id);
40         break;
41     default:
42         header("HTTP/1.1 404 Not Found");
43         exit();
44         break;
45 }
46
47 header($response['status_code_header']);
48 if ($response['body']) {
49     echo $response['body'];
50 }

```

Listing 32 – ./Sources/public/v1/appointments/index.php

```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to process appointment
   requests.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\AppointmentController;
11 use App\Models\Appointment;
12
13 require "../.../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
   Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,PATCH,DELETE');
25     return;

```

```

26 }
27
28 $controller = new AppointmentController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 parse_str(file_get_contents('php://input'), $input);
35
36 $appointment = new Appointment();
37 $appointment->id = $id ?? null;
38 $appointment->datetime_appointment = $input["datetime_appointment"] ?? null;
39 $appointment->duration_in_hour = isset($input["duration_in_hour"]) && is_numeric(
    $input["duration_in_hour"]) ? $input["duration_in_hour"] : null;
40 $appointment->note_text = $input["note_text"] ?? null;
41 $appointment->note_graphical_serial_id = $input["note_graphical_serial_id"] ?? null
    ;
42 $appointment->summary = $input["summary"] ?? null;
43 $appointment->user_id_customer = isset($input["user_id_customer"]) && is_numeric(
    $input["user_id_customer"]) ? $input["user_id_customer"] : null;
44 $appointment->user_id_educator = isset($input["user_id_educator"]) && is_numeric(
    $input["user_id_educator"]) ? $input["user_id_educator"] : null;
45
46
47 switch ($requestMethod) {
48     case 'GET':
49         if (empty($id) || !is_numeric($id)) {
50             $response = $controller->getAllAppointments();
51         }
52         else{
53             $response = $controller->getAppointment($id);
54         }
55         break;
56
57     case 'POST':
58         $response = $controller->createAppointment($appointment);
59         break;
60
61     case 'PATCH':
62         if (empty($id) || !is_numeric($id)) {
63             header("HTTP/1.1 404 Not Found");
64             exit();
65         }
66         $response = $controller->updateAppointment($appointment);
67         break;
68
69     case 'DELETE':
70         if (empty($id) || !is_numeric($id)) {
71             header("HTTP/1.1 404 Not Found");
72             exit();
73         }
74         $response = $controller->deleteAppointment($id);
75         break;
76
77     default:
78         header("HTTP/1.1 404 Not Found");
79         exit();
80         break;

```

```

81 }
82
83 header($response['status_code_header']);
84 if ($response['body']) {
85     echo $response['body'];
86 }

```

Listing 33 – ./Sources/public/v1/appointments/uploadNoteGraphical/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process upload note
6   * graphical.
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 use App\Controllers\AppointmentController;
11
12 require "../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: POST");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
19     Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,PATCH,DELETE');
25     return;
26 }
27
28 $controller = new AppointmentController($dbConnection);
29
30 switch ($requestMethod) {
31     case 'POST':
32         $response = $controller->uploadNoteGraphical();
33         break;
34     default:
35         header("HTTP/1.1 404 Not Found");
36         exit();
37         break;
38 }
39
40 header($response['status_code_header']);
41 if ($response['body']) {
42     echo $response['body'];
43 }

```

Listing 34 – ./Sources/public/v1/connection/index.php

```

1  <?php
2  /**
3   * index.php

```

```

4  *
5  * File being the front controller of the API and allowing to process connection
   request.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\UserController;
11 use App\Models\User;
12
13 require "../.../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: POST");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
   Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: POST');
25     return;
26 }
27
28 $controller = new UserController($dbConnection);
29
30 parse_str(file_get_contents('php://input'), $input);
31
32 $user = new User();
33 $user->email = $input["email"] ?? null;
34 $user->password_hash = $input["password"] ?? null;
35
36 switch ($requestMethod) {
37     case 'POST':
38         $response = $controller->connection($user);
39         break;
40     default:
41         header("HTTP/1.1 404 Not Found");
42         exit();
43         break;
44 }
45
46 header($response['status_code_header']);
47 if ($response['body']) {
48     echo $response['body'];
49 }

```

Listing 35 – ./Sources/public/v1/documents/downloadDocument/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process download
   document.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>

```

```

8  */
9
10 use App\Controllers\DocumentController;
11
12 require "../.../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: GET");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
19
20 $requestMethod = $_SERVER["REQUEST_METHOD"];
21
22 if (strcmp("OPTIONS", $requestMethod) == 0) {
23     header('Allow: GET');
24     return;
25 }
26
27 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
28 $pathFragments = explode('/', $path);
29 $serial_id = end($pathFragments);
30
31 $controller = new DocumentController($dbConnection);
32
33 switch ($requestMethod) {
34     case 'GET':
35         if (empty($serial_id)) {
36             header("HTTP/1.1 404 Not Found");
37             exit();
38         }
39         $response = $controller->downloadDocument($serial_id);
40         break;
41     default:
42         header("HTTP/1.1 404 Not Found");
43         exit();
44         break;
45 }
46
47 header($response['status_code_header']);
48 if ($response['body']) {
49     echo $response['body'];
50 }

```

Listing 36 – ./Sources/public/v1/documents/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process document
    requests.
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 use App\Controllers\DocumentController;
11 use App\Models\Document;

```

```

12
13 require "../.../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,PATCH,DELETE');
25     return;
26 }
27
28 $controller = new DocumentController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 $document = new Document();
35 $document->id = $id ?? null;
36 $document->document_serial_id = $_POST["document_serial_id"] ?? null;
37 $document->type = $_POST["type"] ?? null;
38 $document->user_id = isset($_POST["user_id"]) && is_numeric($_POST["user_id"]) ?
    $_POST["user_id"] : null;
39 $document->package_number = isset($_POST["package_number"]) && is_numeric($_POST["
    package_number"]) ? $_POST["package_number"] : null;
40 $document->signature_base64 = $_POST["signature_base64"] ?? null;
41
42 switch ($requestMethod) {
43     case 'GET':
44         if (empty($id) || !is_numeric($id)) {
45             $response = $controller->getAllDocuments();
46         }
47         else{
48             $response = $controller->getDocument($id);
49         }
50         break;
51
52     case 'POST':
53         $response = $controller->createDocument($document);
54         break;
55
56     case 'DELETE':
57         if (empty($id) || !is_numeric($id)) {
58             header("HTTP/1.1 404 Not Found");
59             exit();
60         }
61         $response = $controller->deleteDocument($id);
62         break;
63
64     default:
65         header("HTTP/1.1 404 Not Found");
66         exit();
67         break;

```

```

68 }
69
70 header($response['status_code_header']);
71 if ($response['body']) {
72     echo $response['body'];
73 }

```

Listing 37 – ./Sources/public/v1/dogs/downloadPicture/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process download dog
6   * picture.
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 use App\Controllers\DogController;
11
12 require "../../../../../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: GET");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
19     Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
24 $pathFragments = explode('/', $path);
25 $serial_id = end($pathFragments);
26
27 $controller = new DogController($dbConnection);
28
29 switch ($requestMethod) {
30     case 'GET':
31         if (empty($serial_id)) {
32             header("HTTP/1.1 404 Not Found");
33             exit();
34         }
35         $response = $controller->downloadDogPicture($serial_id);
36         break;
37     default:
38         header("HTTP/1.1 404 Not Found");
39         exit();
40         break;
41 }
42
43 header($response['status_code_header']);
44 if ($response['body']) {
45     echo $response['body'];
46 }

```

Listing 38 – ./Sources/public/v1/dogs/index.php



```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to process dog requests.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\DogController;
11 use App\Models\Dog;
12
13 require "../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,PATCH,DELETE');
25     return;
26 }
27
28 $controller = new DogController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 parse_str(file_get_contents('php://input'), $input);
35
36 $dog = new Dog();
37 $dog->id = $id ?? null;
38 $dog->name = $input["name"] ?? null;
39 $dog->breed = $input["breed"] ?? null;
40 $dog->sex = $input["sex"] ?? null;
41 $dog->chip_id = $input["chip_id"] ?? null;
42 $dog->user_id = isset($input["user_id"]) && is_numeric($input["user_id"]) ? $input[
    "user_id"] : null;
43
44 switch ($requestMethod) {
45     case 'GET':
46         if (empty($id) || !is_numeric($id)) {
47             $response = $controller->getAllDogs();
48         }
49         else{
50             $response = $controller->getDog($id);
51         }
52         break;
53
54     case 'POST':
55         $response = $controller->createDog($dog);
56         break;
57

```

```

58     case 'PATCH':
59         if (empty($id) || !is_numeric($id)) {
60             header("HTTP/1.1 404 Not Found");
61             exit();
62         }
63         $response = $controller->updateDog($dog);
64         break;
65
66     case 'DELETE':
67         if (empty($id) || !is_numeric($id)) {
68             header("HTTP/1.1 404 Not Found");
69             exit();
70         }
71         $response = $controller->deleteDog($id);
72         break;
73
74     default:
75         header("HTTP/1.1 404 Not Found");
76         exit();
77         break;
78 }
79
80 header($response['status_code_header']);
81 if ($response['body']) {
82     echo $response['body'];
83 }

```

Listing 39 – ./Sources/public/v1/dogs/uploadPicture/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process upload dog
6   * picture.
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 use App\Controllers\DogController;
11
12 require "../../../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: POST");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
19     Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: POST');
25     return;
26 }
27
28 $controller = new DogController($dbConnection);

```

```

29 switch ($requestMethod) {
30     case 'POST':
31         $response = $controller->uploadDogPicture();
32         break;
33     default:
34         header("HTTP/1.1 404 Not Found");
35         exit();
36         break;
37 }
38
39 header($response['status_code_header']);
40 if ($response['body']) {
41     echo $response['body'];
42 }

```

Listing 40 – ./Sources/public/v1/index.php

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <title>api-rest-douceur-de-chien</title>
5 </head>
6 <body>
7     <h1>api-rest-douceur-de-chien.boreljaquet.ch</h1><p>Cette API est en cours de d
    éveloppement.</p>
8 </body>
9 </html>

```

Listing 41 – ./Sources/public/v1/plannings/index.php

```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to display plannings.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\TimeSlotController;
11
12 require "../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: GET");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
19
20 $requestMethod = $_SERVER["REQUEST_METHOD"];
21
22 $controller = new TimeSlotController($dbConnection);
23
24 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
25 $pathFragments = explode('/', $path);
26 $id = intval(end($pathFragments));
27
28 switch ($requestMethod) {

```

```

29     case 'GET':
30         if (empty($id) || !is_numeric($id)) {
31             header("HTTP/1.1 404 Not Found");
32             exit();
33         }
34         $response = $controller->getPlanningTimeSlots($id);
35         break;
36     default:
37         header("HTTP/1.1 404 Not Found");
38         exit();
39         break;
40 }
41
42 header($response['status_code_header']);
43 if ($response['body']) {
44     echo $response['body'];
45 }

```

Listing 42 – ./Sources/public/v1/scheduleOverrides/index.php

```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to process schedule
6  * override requests.
7  *
8  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
9  */
10 use App\Controllers\ScheduleOverrideController;
11 use App\Models\ScheduleOverride;
12
13 require ".../.../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
20     Authorization, X-Requested-With");
21
22 $requestMethod = $_SERVER["REQUEST_METHOD"];
23
24 if (strcmp("OPTIONS", $requestMethod) == 0) {
25     header('Allow: GET,POST,DELETE');
26     return;
27 }
28
29 $controller = new ScheduleOverrideController($dbConnection);
30
31 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
32 $pathFragments = explode('/', $path);
33 $id = intval(end($pathFragments));
34
35 parse_str(file_get_contents('php://input'), $input);
36
37 $scheduleOverride = new ScheduleOverride();
38 $scheduleOverride->id = $id ?? null;

```

```

38 $scheduleOverride->date_schedule_override = $input["date_schedule_override"] ??
    null;
39
40 switch ($requestMethod) {
41     case 'GET':
42         if (empty($id) || !is_numeric($id)) {
43             $response = $controller->getAllScheduleOverrides();
44         }
45         else{
46             $response = $controller->getScheduleOverride($id);
47         }
48         break;
49
50     case 'POST':
51         $response = $controller->createScheduleOverride($scheduleOverride);
52         break;
53
54     case 'DELETE':
55         if (empty($id) || !is_numeric($id)) {
56             header("HTTP/1.1 404 Not Found");
57             exit();
58         }
59         $response = $controller->deleteScheduleOverride($id);
60         break;
61
62     default:
63         header("HTTP/1.1 404 Not Found");
64         exit();
65         break;
66 }
67
68 header($response['status_code_header']);
69 if ($response['body']) {
70     echo $response['body'];
71 }

```

Listing 43 – ./Sources/public/v1/timeSlots/index.php

```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to process time slot
    requests.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\TimeSlotController;
11 use App\Models\TimeSlot;
12
13 require "../../../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");

```

```

20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,DELETE');
25     return;
26 }
27
28 $controller = new TimeSlotController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 parse_str(file_get_contents('php://input'), $input);
35
36 $timeSlot = new TimeSlot();
37 $timeSlot->id = $id ?? null;
38 $timeSlot->code_day = isset($input["code_day"]) && is_numeric($input["code_day"]) ?
    $input["code_day"] : null;
39 $timeSlot->time_start = $input["time_start"] ?? null;
40 $timeSlot->time_end = $input["time_end"] ?? null;
41 $timeSlot->id_weekly_schedule = isset($input["id_weekly_schedule"]) && is_numeric(
    $input["id_weekly_schedule"]) ? $input["id_weekly_schedule"] : null;
42 $timeSlot->id_schedule_override = isset($input["id_schedule_override"]) &&
    is_numeric($input["id_schedule_override"]) ? $input["id_schedule_override"] :
    null;
43
44
45 switch ($requestMethod) {
46     case 'GET':
47         if (empty($id) || !is_numeric($id)) {
48             $response = $controller->getAllTimeSlots();
49         }
50         else{
51             $response = $controller->getTimeSlot($id);
52         }
53         break;
54
55     case 'POST':
56         $response = $controller->createTimeSlot($timeSlot);
57         break;
58
59     case 'DELETE':
60         if (empty($id) || !is_numeric($id)) {
61             header("HTTP/1.1 404 Not Found");
62             exit();
63         }
64         $response = $controller->deleteTimeSlot($id);
65         break;
66
67     default:
68         header("HTTP/1.1 404 Not Found");
69         exit();
70         break;
71 }
72
73 header($response['status_code_header']);
74 if ($response['body']) {

```

```
75     echo $response['body'];
76 }
```

Listing 44 – ./Sources/public/v1/users/educators/index.php

```
1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process educator
6   * requests.
7   *
8   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
9   */
10 use App\Controllers\UserController;
11
12
13 require "../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
20     Authorization, X-Requested-With");
21
22 $requestMethod = $_SERVER["REQUEST_METHOD"];
23
24 $controller = new UserController($dbConnection);
25
26 switch ($requestMethod) {
27     case 'GET':
28         $response = $controller->getAllEducatorUsers();
29         break;
30     default:
31         header("HTTP/1.1 404 Not Found");
32         exit();
33         break;
34 }
35
36 header($response['status_code_header']);
37 if ($response['body']) {
38     echo $response['body'];
39 }
```

Listing 45 – ./Sources/public/v1/users/index.php

```
1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process user requests
6   *
7   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8   */
9
10 use App\Controllers\UserController;
```

```

11 use App\Models\User;
12
13 require "../.../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,PATCH,DELETE,OPTIONS");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,PATCH,DELETE');
25     return;
26 }
27
28 $controller = new UserController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 parse_str(file_get_contents('php://input'), $input);
35
36 $user = new User();
37 $user->id = $id ?? null;
38 $user->email = $input["email"] ?? null;
39 $user->firstname = $input["firstname"] ?? null;
40 $user->lastname = $input["lastname"] ?? null;
41 $user->phonenumber = $input["phonenumber"] ?? null;
42 $user->address = $input["address"] ?? null;
43 $user->api_token = $input["api_token"] ?? null;
44 $user->code_role = $input["code_role"] ?? null;
45 $user->password_hash = $input["password"] ?? null;
46 $reCAPTCHAUserResponseToken = $input["reCAPTCHAUserResponseToken"] ?? null;
47
48 switch ($requestMethod) {
49     case 'GET':
50         if (empty($id) || !is_numeric($id)) {
51             $response = $controller->getAllCustomerUsers();
52         }
53         else{
54             $response = $controller->getUser($id);
55         }
56         break;
57
58     case 'POST':
59         $response = $controller->createUser($user,$reCAPTCHAUserResponseToken);
60         break;
61
62     case 'PATCH':
63         if (empty($id) || !is_numeric($id)) {
64             header("HTTP/1.1 404 Not Found");
65             exit();
66         }
67         $response = $controller->updateUser($user);
68         break;

```



```

69
70     case 'DELETE':
71         if (empty($id) || !is_numeric($id)) {
72             header("HTTP/1.1 404 Not Found");
73             exit();
74         }
75         $response = $controller->deleteUser($id);
76         break;
77
78     default:
79         header("HTTP/1.1 404 Not Found");
80         exit();
81         break;
82 }
83
84 header($response['status_code_header']);
85 if ($response['body']) {
86     echo $response['body'];
87 }

```

Listing 46 – ./Sources/public/v1/users/me/changePassword/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process the password
6   * change request current logged in user request.
7   *
8   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
9   */
10 use App\Controllers\UserController;
11 use App\Models\User;
12
13 require "../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: PATCH, OPTIONS");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
20     Authorization, X-Requested-With");
21
22 $requestMethod = $_SERVER["REQUEST_METHOD"];
23
24 if (strcmp("OPTIONS", $requestMethod) == 0) {
25     header('Allow: PATCH');
26     return;
27 }
28
29 $controller = new UserController($dbConnection);
30
31 parse_str(file_get_contents('php://input'), $input);
32
33 $user = new User();
34 $user->password_hash = $input["password"] ?? null;
35
36 switch ($requestMethod) {

```

```

36     case 'PATCH':
37         $response = $controller->updateMyPassword($user);
38         break;
39     default:
40         header("HTTP/1.1 404 Not Found");
41         exit();
42         break;
43 }
44
45 header($response['status_code_header']);
46 if ($response['body']) {
47     echo $response['body'];
48 }

```

Listing 47 – ./Sources/public/v1/users/me/index.php

```

1  <?php
2  /**
3   * index.php
4   *
5   * File being the front controller of the API and allowing to process current
6   * logged in user request.
7   *
8   * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
9   */
10 use App\Controllers\UserController;
11
12 require "../bootstrap.php";
13
14 header("Access-Control-Allow-Origin: *");
15 header("Content-Type: application/json; charset=UTF-8");
16 header("Access-Control-Allow-Methods: GET, OPTIONS");
17 header("Access-Control-Max-Age: 3600");
18 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
19     Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET');
25     return;
26 }
27
28 $controller = new UserController($dbConnection);
29
30 switch ($requestMethod) {
31     case 'GET':
32         $response = $controller->getMyInformations();
33         break;
34     default:
35         header("HTTP/1.1 404 Not Found");
36         exit();
37         break;
38 }
39
40 header($response['status_code_header']);
41 if ($response['body']) {
42     echo $response['body'];

```

Listing 48 – ./Sources/public/v1/weeklySchedules/index.php

```

1 <?php
2 /**
3  * index.php
4  *
5  * File being the front controller of the API and allowing to process weekly
    schedule requests.
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9
10 use App\Controllers\WeeklyScheduleController;
11 use App\Models\WeeklySchedule;
12
13 require "../.../bootstrap.php";
14
15 header("Access-Control-Allow-Origin: *");
16 header("Content-Type: application/json; charset=UTF-8");
17 header("Access-Control-Allow-Methods: GET,POST,DELETE");
18 header("Access-Control-Max-Age: 3600");
19 header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers,
    Authorization, X-Requested-With");
20
21 $requestMethod = $_SERVER["REQUEST_METHOD"];
22
23 if (strcmp("OPTIONS", $requestMethod) == 0) {
24     header('Allow: GET,POST,DELETE');
25     return;
26 }
27
28 $controller = new WeeklyScheduleController($dbConnection);
29
30 $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
31 $pathFragments = explode('/', $path);
32 $id = intval(end($pathFragments));
33
34 parse_str(file_get_contents('php://input'), $input);
35
36 $weeklySchedule = new WeeklySchedule();
37 $weeklySchedule->id = $id ?? null;
38 $weeklySchedule->date_valid_from = $input["date_valid_from"] ?? null;
39 $weeklySchedule->date_valid_to = $input["date_valid_to"] ?? null;
40
41 switch ($requestMethod) {
42     case 'GET':
43         if (empty($id) || !is_numeric($id)) {
44             $response = $controller->getAllWeeklySchedules();
45         }
46         else{
47             $response = $controller->getWeeklySchedule($id);
48         }
49         break;
50
51     case 'POST':
52         $response = $controller->createWeeklySchedule($weeklySchedule);
53         break;

```

```

54
55     case 'DELETE':
56         if (empty($id) || !is_numeric($id)) {
57             header("HTTP/1.1 404 Not Found");
58             exit();
59         }
60         $response = $controller->deleteWeeklySchedule($id);
61         break;
62
63     default:
64         header("HTTP/1.1 404 Not Found");
65         exit();
66         break;
67 }
68
69 header($response['status_code_header']);
70 if ($response['body']) {
71     echo $response['body'];
72 }

```

Listing 49 – ./Sources/resources/template/conditions//textunderscoreregistration.php

```

1 <!--
2     conditions_registration.php
3
4     Template of the registration conditions content used in the HelperController
5     with the storeConditionsRegistration function.
6
7     @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8     -->
9 <!DOCTYPE html>
10 <html>
11
12 <head>
13     <title>condition_inscription.pdf</title>
14 </head>
15 <style>
16 textarea#mentions {
17     height: 350px;
18 }
19 strong{
20     color: #3598db;
21 }
22 figcaption{
23     margin-left: 20px;
24 }
25
26 div.card,
27 .tox div.card {
28     width: 240px;
29     background: white;
30     border: 1px solid #ccc;
31     border-radius: 3px;
32     box-shadow: 0 4px 8px 0 rgba(34, 47, 62, .1);
33     padding: 8px;
34     font-size: 14px;
35     font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,Oxygen-Sans,
        Ubuntu,Cantarell,"Helvetica Neue",sans-serif;

```

```

36 }
37
38 div.card::after,
39 .tox div.card::after {
40     content: "";
41     clear: both;
42     display: table;
43 }
44
45 div.card h1,
46 .tox div.card h1 {
47     font-size: 14px;
48     font-weight: bold;
49     margin: 0 0 8px;
50     padding: 0;
51     line-height: normal;
52     font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,Oxygen-Sans,
        Ubuntu,Cantarell,"Helvetica Neue",sans-serif;
53 }
54
55 div.card p,
56 .tox div.card p {
57     font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,Oxygen-Sans,
        Ubuntu,Cantarell,"Helvetica Neue",sans-serif;
58 }
59
60 div.card img.avatar,
61 .tox div.card img.avatar {
62     width: 48px;
63     height: 48px;
64     margin-right: 8px;
65     float: left;
66 }
67 </style>
68
69 <body>
70     <p style="text-align: center; font-size: 15px;"><img title="TinyMCE Logo"
71         src="
        gAAQAQmCMAAAACUOGWcAAAAkFBMVEUAAAA+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+P3pJKyijJj9fTjRAAAAAAAAAAAAAAAAAAAAAAAAAAAMB/QREA/
        I94kQRBsIg9Vqgd4B+ich+P60d409V003Fb4TBbR53kmQkA/maqWDXbbfvWjGaa03JhNSfT
        /mFBRAPzHeAvpNRH3otkGYOGL9NVuMoG0TLmutQ3oiWnf5Jcd/
        RHc13hTQLgmygi7mz7x2x32vRcSy6syW7LdF3jkLlS1Nfuv8SBp5jupppjJxGgd4Du0t3u30KsWnf98NV7SFY
        +mcSP3axfqG5zXyFUWA1gG+
        gz3b6PPyW0uJFnHWV6plmomWKuEsIqZ7MFEymg33qzA87cZPftrb9wNcvgf4cp2850papH+
        t2iepk9JdmBpZS0p6u0GHFC7hA3ylnWipZz1TiaKlK9W0uNmc1B3L9vjpbW9//
        pL8W1ozIpQ08GXssSU3TTqfEvV8qeNaqUd8s/OoVbGikPGWGgQAX2Mk9/
        DL45lpLzX0ZiLrm00Zj5boqv2C28BMB/gNsJErKjnKPxSsqRUu+
        J3LKeftEnVDvRt7cUBf45LcgC/eVcMU1Ff7rIqP7GxKc/
        gksnaz7zazgNHdN1LtJYoHeDXMJG36B8/bZwHchfrmUv3uU/E0HU/
        hapdS29JVXdub0QXM28Zr+EORUKyqQYz4wN3gKJk3002de9AtaHrnz4eimFPXBzHXTFMk/5
        GSvSquyamKioTk5buYGFcm700SDMgvnWrDWMjDw9I8Sv6xNs5krMy++
        ZEt9w0da3ijLY8rvl4bU3UXg5LS3FHVg2qNC91niZEF0/E5M6Iqzt7f/
        ezmAaejuLLftykKLHkbESqKnTrKZuNDp1k8fy8SDrrphimZnNMkXFQy/r1GCd+
        ofRWaG0rQ41dKZgE+Z/

```

diJzzRNf50qJBVZ60iHb9o0cADOURBYPh2B8rUlQUiT5HGxqVE33CZlcc74xum7YyQ+9  
rMaxVfvb2WMxnuBLaFaXzS2mDPj//  
B4ePPf5GJA2qdv1qef69KT6Gg4eiKD41LS0iIZePREZXeamKOEpi4h64ZbRorgSYjlIKPWdnosU0peulxo4Y  
+Y1S6B3rEtXYq5zoxayYMn2JcUhsHhFTn97ZPTFZs9GCrhmUtvPJ5czckjerg+  
hefL10FV30McBIhweiKLGm4NSmgjiViw6p66Gn5ZWAbезEJ4XQh2Lq/  
mDvTJsUx2EwLNU5IQnh6nCEhuY+9f//3W4tC7Ls0HvVznbN+  
vkONQISungtWZKVUN8jENn99gEt9DPu0Dctpf1gdsMqaVex0GONoxe6538Ez4TvgBHpu2d1Is0sS  
+gS9mRMpduj014hqjR/HqebB9so0FUViyQW+  
otk8nz30GniZmWq2L7tJDcvIfykGs9Piz7DTeEQ0BtUZJzBm4kVuvMgISDZfoFwCF2/2  
1W7UBLEM2ih94mMGb/uk04U121CF5AgkdEyoGbaTw/  
JxJOKb6pHTmbp1CTbM6ELqzPbFqzCRIOLPSewJtozCyDWQ4ELQb6ax4RMGTxvqE6ArvBUoF26L64X1c78Hh  
+SiTMAhZM82xcSrZUK1r1USPbgrA0qbmFTjy0q+1QacF7XIdA0FqgkVqnV363B+  
Vx0jreRsZIKKP/TkKFC1V6nIHH8/MhYRiw9Bhj8Uk2tSKjQI3YnAA1YKPahX439+  
gk44d2pTjIb19g8MEd+1KCweFlK/Znu5Iu4IYayQd7hYTzyzICj+  
enQ8Jd148h9PCCZI23ZEiRUBtL6JM/  
JfShMzDH01sJYIR7nnKvgeCJg897cm7p198GqNGYDr3gKQe0jHyK3vNNEUL0vwZb93BU05xW2APOXd  
+kX+  
FNZh9dIYQmKjUAwTvJHBuFRUmG5Jgm0ADGFB1Zbulx9LQEda0yKxsnwgyN3lze7RtrST5GL5ocC4yv  
/vSr5/sR3Y9l/NzLqpmQ4IJnwhWGpmvWjQcBlyqjNc6Zdg8mbqHPJY+838ZicETM+  
X2ukVGDRfS0BMcCq/7C/JpH1l3IOwnj03SP+rZHhWiwrHvjvd8N+  
QpYadG010H0Ne6li0zNy5BQvutszi66YFwqSoVwkj8EXsm9FyRRx9dS9xwv0pWqoJzw1TrrYPQmEujSGNtkr  
+75TgiZfyJH4YFtMsOsFBEBp18hy7u/WHEG0TNSVQnSKfSHZFKmMVPpIy3LeZ+  
ft2VKPfZt2cmG70spG18hjTnT6dVcmhomdDvKULj33fGeb0T/8B65Qk5dYTEWjs640Q/  
dOfKsb9KH7UPjghnzj8x7PkAax1SJRrob39LVesrv5HsWoS+1/  
ODDBGmqF0BNJIKSZvQ5VrvDPS7dM+3oTeMrTFqx0N5otWnS+  
ajmSc4Nm4E2kZdXjH6UwxExN6jBoXYyoF76YzbIEIM2Qs06fPxPXX7otvwrVqB0TuWahhCj3a69  
/+7DPvnu+BnBQkcyabABWrELu0N4M/  
143LkeAdp1xTgV6uFnBlF4tD0B1lztobLIxza18gbaFvXuZsM2+  
GRjeMTtzt2r80Uh06RpCDx/MN6K0dMo/  
VuFEqLiNHdY2ohDlkJk0i6FNrnM7xPZ2m97Fmlhs/q35NU00TbGbPTIDLXiics6vGEdiI+  
mV01+M7W2D6KeokIAHcC8GkrbqIClc+dPd8A8IKFbo4rfEcZEGOPPaQ0QN0/6gL+  
m2VqFMunkqdDirSG4LyD03Rxe0SQ+Ce1lMrqRm0ecr1x5aE0e30B+  
XnA28uCnUKcCcTluw2bVsN0Gkvmn08w2YoWJtpKgz3yWqsktEEgb8dY0uh6wF07c2UqbYWf  
/j4zo7B0ZEKUQS0oQ+NqwnevdonaiG1hlU2i6DBAu5fN9ns1wwoTdo5+Kcs+  
yOYGBa8toFeDz/NVHCzmx8D0t5E9PG81whVi9y9+hGptqamqpHr9v3/zPFqiRRz3/YRSy+  
EShQY2VaE3LZ+3Q16NNtKnTvCGy3n+zrkH1w6nywhH18tgEBoKfdicSPj/X859x5h+  
dv5POAdr2oTsuBY+6  
EK3aXbHxT9paeqJCg2e6MYGB3zDGh19I55yJJjoMx7ZV7yBm2Cv34NKZFdlkzh75GhqpNoY  
+RyISwdhNUX7Txz2X3/  
Eh6KRLFF0SvhACnTPNV8W45BFt7jIFhP5FJUUVuq2KMLt9sVsC1Z6B66hK7wAR8ZCT1HszwmAVwNM9k  
+W80hsg+sUHD0Uwd8Z10Kgix5jDpZ3/  
HUKuHF7vKx3NrQXsKEUUXyL5Na5T0jd9UgYwXcfk2RWIasNa6TREh74GzjrqbRI5pSiAPoaWh3mbtdxp48UV  
/cZs+Rzz5MCszZlHu9Vl5EfQeX4IRXvQHEJv+  
faG9201JlMQRq6eseyqr6kpDZn6AxT5Y9ezly4SnNm4+C0/7/  
gqRhTsuLmZsQvi4lKBYH3sjNKM3A8Kidpee4iyBybbwSNALHpe6Z4fgFQ0jycB6reIb6ju122fqWtsqkBwJ6  
+  
GUtcZ1sQELAb2qLRlHrheJ6ojz7ct0QcYjBmi61BIUzeUMIf0BSmSotS10tg85td4SzdPij0ryGYN8Tn307v  
+pFthj4yAjBS82r/zHLupQbQIParcouGTqxTW92omfrccUSelLbp0qF6HXA/5  
kPexMwKzaV2y0GPBUxglMj4j3fqsv6nnyBvweP42Av4cQ+aV+/  
zo1jZFIoHdHSQZL8jpa00wByadw3dPahfNGVqJmGKXEzq6GtEs1RcV85yzq4joZR6GuztsBQ  
/MiZi92+zXMyqQ8mI8BWqnP7LuWpE19udYPf9A5BKKfH8keAHLjmnHUt/  
jqtXtX0tCr5EzMT65RqWJpA9PQlToItss0B05dhxTpXwAUR7emuPXUoNWoe/eB996p6w+8/  
ZWe/6r4y+wNNr1hqhaB91JgI91ghpx7pXu+Vs6jyAcPJ2VBC1BdLwOMzq/3  
POXV3UJwT0DizerR1aRMXgJXQboQOHKuS6JOWrcRYdHV3gW5Ks5B5Adre7J8Dx/  
z7cS1hNY1ca8uwo1Zo7xd7zEKARs6ww5mT/14  
vnrSAjn431wWZ4mX40TfEaLLgGFqnu2Ux5ogW+yBeHsgcXH00h2JKg3TmboOPMPH3+  
vGnGGe/MudykS6qoNgPOTH12UT2NyrmkxoD53vY9XdETuIyCokm5s0ATAvBRXuKF36Z6/  
iJAgHvgbaRwnybG+PLbuPfvU2eJpN4dV5xVI9jtnFIY0JkFby4ys0E22cAUeD9peNC+  
hE4uKV7tfnDoHQ3K3ry6TS3VcPytw1AxDJF39r3h09PgQWQQC+

kuFFuXWC93z15AAJyPVjNisctc2cGGN00UIWZCCDhVId8eMAuHskaWss9yJE4X0cTb0riCwK  
+09cviIzagnedG+u4z+hU+qQzy/Qe4KW0zHNY1RYyPMexoJ4gEbJu2b70Gj+  
cv6ny0D1BDZQEqV0keRKuG+  
mg9hcTZyqqWlwX0yHZEzhTcBSb1m14h1uhG4UQ4hU7k0HUmdn582+vOMym8V0+  
VcXw8iC2FMhw161rICvryL010KFzwwRRk+fI0/Rf2zkQ5dRyIoi3J8o6xzb7vayD6/7+  
blwBpWrIcHswenZpKvYxMEihfS2p133b8FhxNGdEPK1S/iMsc+Lczet+8hkP49cyI9+88  
Ib6JhD3RJ1WfU0VDqXExB8SWGvcYF5WeQtCNcaoIkxEwa2qg8FRRdiBBQ1qKJ51+  
I1HvIj3VhZ4umkKZRDk4HL8DA9k2i80866oYGLP06PZmI+xLLd7Yy/Ekq0IgfQpUCu+  
V7VUG31W0IDZfd0+CPfLV7Ce02RISkowXMzIfDAfEUoLQrF25o1hJ+R4BZU6Yui02x+/  
BUHsmoi/  
BJLUEzhF2c0YX7d37eAucYTCK0gZmWE8gU45dEWsQewZVokE8tYeatzXuzBzahRwPgtRLDzfVIs8RD  
+  
rUhiDM0YqXqIOQL7nTueBScf01kPW4XuhlVmpPao3Y03oHEko7asDSHEzF0m3Nq10iNpZHsuxldbADsM7oXN  
/G3UndGinpMEVvOUE/  
hafPdm1yla2mYHPxY3nzseBtuJ1DFMgH0zo9dmjikR5J01b0iwaIFRKVCZTX11EkuU7QpFWLH6Fs3qkBTxdG  
/gAOJdceqJGq54zfEYluxzoYSWJ4bZXf15xfZcsWz0hstZF1jiAVRLU6eocouQnqb/  
JGktIvK0+  
UboQVPWmEvHXSp0s2GEfbMTj5i81stN10N21ZiUizAzLmquYoIykeBwPAxmfCOe0aX7iVXo9hXR9rVK9Y4ya  
+FfRPb7FvD0pivaUjJsyI4WGEI4NqjyqEcY+EMUG2E0EMzG6Ci+ZWtF+  
zsMr8ovRgBQ7HA9hrLgBLpLufax0Gsz0wWqGPK6d0X3ydbA2yWzY4A39ixqtqHjx5hWGFUOPLa3ohcXuvF3r  
+  
soNeRdnBibEIFJHBEqcXPd0x0PYcy4v1R18nftmxk51qfK6BNSwjFQ80QDHfax15c9gNCN16FWeurV9s7q9A  
+  
ls16SV05o7HsN9X8YnBBY06q5Sm00tzYK1bRHA0Z0fo4hKiTg3gt4nQ95UNyyef0340y4guk  
+2  
V3x4tFCPPzNHeSQCsRS3G218JhUQd40iTRTnqbdGV3a0Sq3oshF02V9PSWcs4Xl6577ccPcsfF3rBoZKLCsW  
/qrtgmXirLQRSPm+r4Ffx91DwROV+4doJhVPTQmmBTNN5B05o5Xk2XE/  
maPlvXoWTrVEGPG4rZS6sV1eJ0JdgMYbsKtZ83m+duGY2MY6sBsVIWuwRQ6zRpH0vc/bq+9  
vUARDnSYw3U8HKIv0zZRvUcy29EZwt7Y/2iKKGxa6cBN547nuA/  
ybnYLViWE91RTAoT1m3RaHRLs1K7sU0N0ujRmRpoL4lemxpXA8QSNbPZtwtqUd6PLWCGzkb2irq104090HJ  
/uRT88GmcOKYUyCdfddJudXJqM4wlw+  
jSPezD3Q8wrUuP6lroPuz9FcFyCLTeuQ5s0F1X2iAkVyfHm6BxaM3cYdJsV5+  
hsNVSEd99MZKfMyQFg6lmKVvxCUYrag8PCBxNLPnJjn40KF4ZzPAmjJSQ13NG63M9epWFRbtRS8e  
/c33ej9Cr0fVRX4sVgU1Wv7mvtWiX2PyLF35bliidUKQG4vxor6+  
mZzVy62bo3w5qKyhQCZgp9Bsx++q5KBgY+  
lqOTbMVIEGxc2qvjecakAxAjC29MCzNoKQL1eKG7cd2fmMEqNmrPgNmicaG8SiBWytxXc7pcjjoEb9ejINunM  
/NxSncKhh+e3Tunbz5fFp4S1bYa2xxcWarjBfg7KQrtaHffqQQJlcyUvt9kYFKSxBK/  
hdE4ikeMYhdVZ1CSpsZFt/UBD8kTI7E3cMDCVqS5TMHAtKIMvmIQHM6i0lWIwape6EtRf4r  
+F10ZI60UHI5XkERzkZ+lgHA0wB9o1WJf4e/  
pJQB0iWxzVKbH0ULTyTGakeTCYayvlrn0V6PLu9upWoTYVD6g2ElfBRQptlaj74mRYBtlQ4Re1sbc077CcY1  
/mmxH36HMPeyBo7TejI/  
DrAtlQn21vHhLVHnjjbWnevsA72VDV0mxZGo2vFEVgfu3ao6G4ulSXc43tY0yFvm7iqEEJDscLMEjonLds8  
+Ljv9116FAnn8x+  
XoXLBySe5v18JvTeCnfwnaWmhXavMfN1V0W9I94WHI4nwZpQZCbhYVJFmWA8rfqcaoayYH6sdI7Aq4xhvBSX  
/7ylIsNEVtmg9ErRq2ieaJfILpfWU/6  
KRkSGTzm3ojtczYK1Bmx37GTktATPrwESEiSkYQaN4S+CmX4TY30UFBPW9gQt41K7v+  
U0jyuBOPCWuu01w+4  
AIIWzFu5ZMoAxY9dmbUGN5G4A06txGDxy0120nMIGcy8dfezaMYjTQ6S0qfn2JW8AwQB0onaABVw95DMoHCO  
/Ueg0EKZ1nHf3jpfkxTcXDIZleTW47ku0FzALJ4e8wewXa0Nd/  
QatyJWVla57M5JHWiID8DNAkPT6N3UKbJdmpsbtorTnfbXQAWSbnJGd50pNskgs1niiYAWnZ6TezJ7D2DoTS  
+GItw0M37ABxlbqUz8awmjWLSrewY4dmqYRokPHAppGHA1xP1AMK5QTr+  
TE930bvfcW8roKnbcOUQFbtOPjM0nc3AlTXsFZLeifKdLt19orC46giAAQopjpUS2WJN0gPqY0Dnm1hUptmI  
/ePf/u2halw/03tnxPGbt937lg4VB1DyPU0oYBqeloxTetjvDrTt5u618R+  
szen9h3KNuFY2dQvAkD3p5+/MF2WwznnYkADMnc/vDDRrliBr+yjCIik1w+  
fl64zNrXPDELxmGK7oPeAMGDLpaVu9hpyxES3A4nsVsADReJJXXcRhESqlJeZ9Jjps1W0nTJM0j5o2M8WF91  
+vsmAXex68Jm1rQmc5U0B/0JKb6PQr7vLqTXZ4U7U1YfPzJm8N4R3cxtFcqZfw2gl/QtL65  
/xlZmKiHmbqSFsfzMM35X0WtqtudA3SCi8aGjbsBTN6m+2aEze9HJVE63XSby/  
cFzpq6QUvcXpumN9B725eLcr/sAfD7v763yD9V+  
IvfVgvjAKtjcN1Qe03TYVH7ebZCpRHMjqfpmDxp4y4wn0TvmFk/



kbETuuNPnNHPaaviTodkG6kr8RnH0khX517X5tQep+  
 YwyuwYDxYSL0fTmygu3qd5wzfGNHc2DZT478NBLstyMe0fD9seJNY6H7SNq0eoZhc4VtYhOWkRWc7aZk7ojj  
 /bCwaGmQLao6efCoRdYZB5v3lC6SDdfNlJJZU12Ax8wVrECfwV89WjVlaz+Uj+  
 ubKSE3wj15QwgGSoDLw4vV5gETui0F2DQJ1odGhkzvUVbaSzzxjem9BIoDfzxUW72THrzlKiQ  
 +a1HtWwK9++Fcc58/7Hp02ryk2aewAJY1VF3P+UuhnrF8wJ3fEsZu/  
 SeKTJfBwrg3gBjE7pSLy0JrtPzBMpDvuAzGAf/2wuWihzQ+L/9huew2pXs2rPhufR5+  
 PAUwbHbb9zDFV0WC9j47yxCw7Hizmw9p4CW1VJSqZ0QlRKInRBk9XMg/  
 zhNY38ek038y78p5v8c0imSonqjsfFyr8kHhVKx5t3Bt3Y25Z7AGk0d8DheEXpY6MvCiItZZ7jN7JLJ4Q4cRI  
 /cLgrmb8l/W+afSufLTCnxwSUzQFx36cXi8hAzYqAfV2TTyUyo3eVBQLdLsVAq/49/Ko5/  
 FDRowyMwBCuudKZfV0TG6nRy3/fExzv6ALw60sXTxvJ82MWzssX/8zK/  
 vIHRQRkc910Ahm76dGkeR7Gart9Glwn/nYbcMyVG//  
 mPxfFPgblxSJMeTp0tgSXs9HAyryg4lpb7HrFttQe1ZbpK/w8q/4SBfMvLxak/P260H7SH+  
 VoC4p88ml0Yb/uzTXtfnTUY52ryP/lkHC/A+BV4BtnWug/  
 doB0JtGJwnPRjZbBJADhnnEMvI1v02mqZ/43  
 Mv94I49L35QeJT4c1F93ch30Y81b3NuPPqdBLLYPjZ8M4x3+/0  
 DcZlC5KbrYQF7rFG00jv6F0wr6ED8pAfXFfwUbIcYN335YZdsfhwppJGzVV7qpafjScAbD0cVcUxfthyxh  
 /4pZbhlpVeb8BF0YFynzSD6ilgsQsV6EMDuWpsZ15d3uC1f9nvn4Z6m2/g3124E1  
 yN3gWitJ0j8kfDOPg58PgJifxnj5x0+iRn+a5taJlJL++  
 TBrJZ1kLchrAlW6mTEQgPHH3rWsEqSP76heRikTcUM1dA0fQGQfxtu45+WPh4G8iQdZ4Y/  
 mon0x0Z0PgdxtwoUSWd1dGU2CxBNJKvRbRPK7djYrcNWdu9ov39mHVkHSVFSiKOLjP74fCOHRChZB88nrqm5  
 /ndkFyp2LQJNNhcbu76PKrc5xGfcfXw/Ew7yKJSBt39UUHYDpAUg6/MKr9uRjJ4aI+8  
 TVU88bTmdm8hBoZQ6cqtTDRK5z+9  
 nWuEtV1U08ERAbvZtpZQ5pXdsYgBnJfNUE1zMuJpW02t1AEE7WVe95vgFQ0vk180zHLZCIdZ0Sszqlv9  
 +vHAbKgj11a50d6kNzLYjMOE7pbF3n+FPhB2sypoC//2  
 q9CadPuQDU7qXmt1JxiSYhF5Husqr34TD0QndAeC/KytB+sQuvV/  
 rWV7pxh4DB0S0lYl3aipvmazB8SD2g/S+E/  
 rPIy2UnckTcXc9kVX0QMPscypW3z57AqWiwbLjbtFnKhBcxsyPxx/WHGd5A+  
 CvLxXDrmWNH6KKtd/jlxPDLGbYAT910n8C0dMbQKXg+GFMic5fPkg3e6EJpaJGtW/5  
 BC9qGXUc6TbzvCCe9Pvv0cQLjiks2f+oUvVhBud3YeuIP1nwaCjang+  
 V8PGSR7sh5rFltdnaE+A1lm5pS2Af7vt07+  
 Xig7yDXD8LPQJvTmfjyN1IUYe1Ba7D7x78dVcQjJ7J5K0vWCWmc4B0Hez+Z+4SZ+7D/  
 NnofsEjxkAr00LQnPgT3dhC5XfVH27V14s/s993/  
 rLb8H4HBT1FCF437STBoZequetGT00fB3700az98MHEpxt0q4iL19hJS200t4t7i8E/  
 JfBh5wIN7CPTp/EgzWgqzbL8UijEOZBRnn8DRSazISRL++  
 t0djINz6CC3A8ZdCjGK9sDnwgfGDYJca8RvBl50DB+iN4BVohquYXwWl4jXv+  
 rffzUfXeabtpe/GA6dzIuiIAjDbL4fge0n0VeistcZ5/  
 AajKwXsTs2ihsno3zt95Iwnp7CoPPXx4kTud/NRxGg8Fy3Wh12cd3boP+B3t3op0oEgVg+  
 FYVm2xuKlgiuCCI4n3/t5ukXYpFMiYS00l9Mz1nZpJ0d5+T/1  
 RRBdQfU1qLY1b5JOK7n4hTsQELtss+IiYSvk1B+HaU72WiH9F/P/  
 JR6L4BbaIw0bEiJNTTOSr8+D9HDC+  
 PQc150nPoBJaQWqp8yP1GU6AtFy6FWCZ72ZMKr8HVjXgrZAI0YzwpAgosDRqN8bx5DLSeummj0USoq4yLJn  
 /V/Qv/DgCAGmuziwgjTe2DKBW+n3fuBTothS6jYh6P1ERmYT/  
 ePAkyLuGsKmsKIzPtSfj6Wr4ZjV8+7fp2DQNRZFpwxzZRC/8BKJECUqorSsX6cVZdA/  
 gPC4Vv+kp/  
 eoh4gRgI1UX4uKZdrpCZ3j4sRR439Vxmyrm2LV2o7Q3yBf7ebTNZrHv00nSD8Mw6P4ThG+  
 SJPH90M623ny/yAe9dLSzhmNT0ZXkn/94ZuF3IJBulrPzM6MECnpYI00VKKGm6xpwyf1rW/  
 UJNmDoU/gpx8DhjBpj5f28o2XxX4/sDVVYvhJTFJ10+j78dbb5710544Nwn+  
 BN6J34VtRWEoanoTTDx5Hd7yeNRyOp2Nr2Vt4md+39  
 WS2H3gbC97RLy0N5AHilWoCAx6rXriyskbvf9JDEtiqVEsXGXv/gQxPf1f+  
 cTnKmGGVpNpBcsi8RW9krRTRu/AA8kjDi5hSaH7nKt073SAIqotmur/  
 oLclXdtcpwHiv11JniP4UHqp00TK1RoP59pB09Q4rp40M78EQy9mzjt5NDtv5ILWmFP4hInehfQSG0jK8mIN  
 /BCrjGfF/8dUZ/KVQZ1QACvq8IyOf0l7Cg9TSNxxw09ybOUExcF53yxiyYvCBM/  
 Py9Hw1RKm4eUVolRjigbovFAC7DtYwvE7yTQDyhaEUYLc/2FjQHTV+pfYbPyVuTfaZEXYLZ  
 /gY5d5DJ9tPrFPuonahLRS2WDI3CFxEyPBm3Q396u6eMlzm24P/Lp554+/vnDd0h+  
 lm5gSa90Dam3PH9+E926RDKmoX2sL3y/lGGi2e2MPwdsH0jrtvqWK8UxT4XnyuLls9z/83  
 ij+h99Hd93rHQxGJqF24E4Wl1VfcdMIL7H0uggS++P34o01kPpC7vcjpahI+  
 0Unr01HPFU07cC8CioYlB6BwMcNPpe7Eq6d9gvwcuTma+y/QeLn2+  
 cgUsQv3oJVhm4WbwuTd6CPDz5gsn/  
 Ggw3Pk7iALnnSu3oSvy88GQxG70Np6n0a5QPh479c2fz/



cctI68ZM9XHq03Fr4tvpCA3mdpNr+whKxC1/VL4UcW0CLJ60E/BnxEsau5/5MT5eSU+  
QbRldfCCvY6ruHG9GFK0Lt+MnIHHqxAUKnDHM+4godTTdDvtOPNt6Wz/  
QWNNm1LMM6eSYwnDhaK84W2/  
G0pqTD8U7YYRP125JWBBC9SAzZTqdmqZpGMobWaayYpjm2JrOr6b+DFfpx6Hc6MW/  
LHIeux73FDGwC58rfYAXzHYG+RAI3EBE2ljFcAg/iJ/  
atPa6vzNyHnvXWwMAEaOLt5Ye4wVjkr0FCjeRJ1ptUF/CTyLHodxR2a+u/  
PL8q98zAECOLtyCkC4esQBtV0cpULiNEWPFDDH4M+XdVPu/+  
hch57N35EMRZk8JNzM55Pojsk6cibypVdQk8Gq98nWl/KPIzpm13onXh/  
xHoFbfSdUM2PnHCsopFEoGfQeHDv5ZnXj0JPsdwhMjo0FJR8ujzsgCcvNPHQc/  
v5Mup85L3w7TBslJn2HLQ3hq0zxj+  
kTLgMKtCNAZFqzh0eSJ8wcn7NcwJxXnJP51hL5prnWLF2EPZwRuR8FBbgCPR0XUF5UXMH+  
kiIv1P4soo+gQaMFiKmtwDaEanrFoLjXP3Qmphx79zLI7lXfxn5+x1OnxWhYX7H+  
SsVEvFQer6wl7fCvdTtUY5Ib0BzmthT4o5NaFx6CKu+0giuG8iiGiGrmyGNf/  
GqXXKX0bzK6mruh44US213Betqmjd6TcfumE1BAeQRlvdBQ+oG/  
GYm3uL5FH0mLl1cXRmEDNSsIjScJNqjv9MRCoMvqIbCFXQp8Wokvg28nmpI9iMP/fcb0/  
MUXrf4W8rRfBUNPp0SbFI22G4WrLpBwoVMghvpsDLYXuanjhwPcixjqTR0W3tS5l1jM90CxC8EwLKoeFboJNC  
+v4YWDcuHHnewYsDfCdldtimp/  
MsN7NV2LL7ZcjYCTNRwgMoEq28SZYmk1Pq6EPz4X1QSmGPn/Qqru5jJmo/  
LOts3hpgvB7EZgG+IFlgQqXNyRlWgwUSpQET/ROAgROZiIRm007+  
Avg3YUuBvMvDusLVwF42nd3Cl/EX83+AYZ+  
tXSyQXb6GAZ7e1S5FN8j51ByGdAVG7k9tI78G8xnYjC/Z1ifLaci9d+IL5E1YehdWWy7SEK  
/8gUj5KQe0HPnfSyYQLsIASDDQSAG83uH9TAfUpH677NT8X+wepVuccDPvHLoCyzoy+8  
REgpKggVa+405svNUUXk799HsxAz+VyEAIwn/n+  
pCmTzjRTmuP5BNAhdLLAonBACq63090nLm08lBzNnba/0  
wETP434PAAtQOV0qGflatkChQNu3wqLa9zJ3yTyCyhEVssxpNeiEWaXK7mbt5K0bsLc/  
gFy4AiJvhfwECsMCaf48wyhsbCxguoYzuC1ll4QJo4cseKjt0eq3A0YGWUACy88Q6+  
zekrnrIt409/  
oIyH0ssnMZgFKAAaRigS9DmdLnZXUkNoGCHdaw8n9Y0NpLHpV0Ji7NuVZb72SpLFJ/  
dRTqnfeHAPRUK0XgBcMdVKy0c10dhmG4X/GHI6i0H4tay9zo+  
eLSnGs9deb3FJH6S60w62CFbwLlFckH5BIFKkaXvFSMBmxyXC2/  
jg9W2kp82meiDn7d2KIm0SmSP2VEb+ayEWBChe89IYjFxZ4JGkSm/i60ealex/  
Ux3DZTubjfyhC++  
rrcvuxSP1VURhUW4xkIADl0psvq8nloxIidvwdlZwJnexCc0QtpH500ui8CDd+Uqk/  
pIIjKud7AkQKKGwV195ulCTI+P9xgdpPbwbkGGjZNVG5pGNwgPZOVCK/oJqT7JII2ufN0x+  
FPqweypdVZHHUlpMDDgbMGRYw5DNlRYy34r3xjycvhWvp6DqmrS5EroZngpdA111mnlXZJwEcaLYETHyqQj  
/dnvspE5jcrQQtAYIYlSyBXRnT7w9BhIuGJHeM6zihwbh9ZJXOMVwDkvttjxpmGwg/  
Rs7G4W+6  
lEDADLFF3V7Ia8Wv0FVyz4T8f7WmYQoE5Q2TsFDtjiFJ04R6UgLkVmf8ofWsCEam/  
DgIZlqnG1Rc048X06heBDNXz1TdGuT2GAnkjFZ+Jmq3vy5yCEolJ+4/  
TPQWoSP1VEFBCLuAvkP3+  
LTbbLiN1sFQxZPcTZTSh4fWZH6wJUQ1Wo6V0zMnc5H5U9D3VFyqvwWClOZlHshN1/  
EM102XAHYS4ruQOR3Xi2uFKuZq6I7p3WtwuS5ugXsSTM/  
FqtzLIDCQsIR5QMs3yao89FHTV5mogXo8ktfHzlSbUyDQFp552hWZPxHWTcWq3KuoX6azDGTgSrfAWo133pz  
/MmwcC1W5V4DARpiRQyOshTHV92bTENETUVkKM01eaoqOyCkzZfBTX1xRuITkvyxmL+/  
BAJDDSt8oHwTvRj69JZbb7I8VNTtYN3m71G0R0ZPqhPJYv7+  
CijOahE1lPATWTj2QeipioGGb+IkwmARbdRJcQsDzCFXUXha6kBcqr8CChlWhQoh/5  
bqKG0YE2gkmwkeLVQbHa2L0hjasRPPrjw5ewdEzN//Y+90lNNGgjAA/90jCwkhBOK+  
wYA5TL//2+36CJIQYiMEWVvpr+  
JyKrFTKZmfGY1mur89hZAvBV62nFyjjlvMvf7qsPr5WXcfs3svYvHtRSMpF/  
sDWAFfcrug6UUPB8INKv2vaI58lGf2Zan9R9BTav87SnU3WzS/  
csSMY35q6eQi7Dhy0YvNe6hvFe50f8xnA7EN6ewcvi3TAYm8hAW+  
rxtauXWFcprQ5DFD7JdQXxvCr7Nv8c++mZu0sf8adtqrhRKUdaUxc+ip9Jf/ZsjLA3+  
Xc4L5e2Qa/C72tFHSdSRs6g/jmZ3KQ/avrM4/R8mnndM/  
P22Wm2w6DsDljVHbL4kU5di08svsP+LdGLwhWE19Prm6dQivmmpVL7z6TZaMn8/Vsj9AuV+  
t6YuIJqUgMn+RMWP5bmmS/P1P9f9ZF1lkcw85K+4yLCFytZUfZrjU6RUijD60vj1R9NM0/  
rEP8T69V/azjRLnA680U7ZzReF/t5nnwLUPj4GPIGD0AdV2L+02luLGX/zP+  
i3Rxygjjv2PWQRIi4W9eHA+8z50WBjiXIUON3LcF4Bmm3guyK/  
bMU8Fpj1lrXmFnXNLtrNk5LZCkKizbliryVBcw/7s3qJf+  
bNHak5tWg2RnIotwfpeAtPksta/c956dxxK7TWPcHi66nPNBJygo4V802rrRdcYJhZ/f5+2

m5nLfXMPxXhmYet2VQ/3  
MU2sNEr2vDcKaj00HGyBMawxfT7yoTSXWXrzJm89d183ZJdb0sM5w3ZTivGLtpSdL/  
EEJqi5nx/jGNmGuGMZ24+6mP5gFJqW3v2b5s/snI5vv80fJk0BcyqP95Cr5xNT+GPd/wv07  
+sIU03+  
YMYwDQ06DpcK7GCPdQUE1ZbK8kpymVpv4EQsg5I7C92LJmZz0fraCQFG97zzZrUoRDftIn9fuG89VYhvNK0s/  
+XsVnY/zJ61o24ZSSBjobNKX+  
PKRdP24oCugKcWiKswdSE25J10gID2Q62xCw91p6MMkJLzxBZ10usvXaN6bKEgp1Heys73a5i0pCf1UhHkcw/  
/dZTSDreSrQcF+Jrnc9rvTuG886MRcVN1jKoPxGhpfksPL61WptdNLEzIZ0tosCDosTNVJ  
/1raTTm5vNud1BQQSzX5PhvPqMqSVJfxaFts1fnP1rHR+81Uu/  
kY564DQ249AHTpTohXw76eiNL6cJ7kvxabu/ZfFXiHpSEvrRsmduhj4ARf/Cu/  
bGSaa0Nowm0drf+zy04uYo6ytJP6SeiDWSj9A57KEYAt6kCfJfw5E+D89BeDUSbVVCV0o+  
jCugdk1E3QocPmAaa07gbkxVdSboPldjJOnfiH+PCQzEEbyyT9r+  
IHnuS9Geg8BzQH1LUx6huc8Li6GHL7tbDWT3kTBAdHwqx3mA/CYJgEux6gEIBirCU+  
hJpmvXhr8o+hAheZPoe/wjdM1LXFC42HS6dQfrnd5a3S70usHtpCsAZvfdigrmnICp1Gw  
/01pf+bPqxb22ken740V8tlFXj5Eskm0Hs6nNog68RGJ7bjbpdnJMV4nQF2GiLtUff/  
kMtHZn0Xq/m8/nu/EwDGz+ULmWD2X6/  
jjZjsdDQpYCDskkbwfBNpiCEHtxcpJejom07IVLbGpwx61Db1X3LNMkItPy6q02f2jtwphz1fYGuzJ9f2LQu  
+bZTJ9xA3qq5rVbW9VA2zzy0Qn3CMV3P+zIsbf5lwqyZjTFMC78sNGdEINzLxItM23+  
ZvVj4b1b76FQq6o22nGd7FAVrxgl9hTyj8GITfK10K5xNOWt+  
b9IV4U2m7R80G00Lv4fqg6BKUTEaEI9CISdoC/  
mGnZYEWI7ztBNKNyBgH11Xq61aN52UYC1rFLU9U40qT+IQsRJU+  
RTM04x0qUoxpzhgFAcYSSdkD9p7lsoxhryU1k6zzxJ+kMQjunwmurGNL/  
BKbPUTZQa8qUAdFfLVmmR+  
klzi1CYN6100Nn2JemPo0C7nLTBjaSPbE7SRxBiZsgXTiAUpbAwWLzT3MRd2hWaEemmJL28bNuVWvfmuOKNk  
+pnC3UYWu40yauZSn0Hb5xuT9dntOp510es/  
hhCEIBfly9PxsQyihSndATHvi0XN3NswCXz1JNdkgHGw+  
CxShoEF1XpylBRTKIPjV2XKk0xA1UeZ+bnYroGTZ/KkWXUv6a42/OJZCQRsWv/  
QJ5RCsE1fGBqIkHkZDTap1bS++vmj8Z/G4Puvq2UVspFGPtK/  
QAUkw5oSylcKz0BV3LxveSCMsap9k3b9NPnKDDAxRiJo7GR8694stHkv0zvfmgsiLV2Uk88yBKIYw5TW  
+QjOyH8xfkQGIFtuEUHs/9huT8bP2g/  
qImDtVZ3Z2qluZLU1SbIC3Vj9n64PDlMSCCAXusoqC0F22PRwzZ+EtoBV0XtAFFG/  
NAsNicoXKew668JygiKKGQpQ8NhmB8CCEUXWqa+qmHHIphTDjSOMrP+  
nkXvRMVEhTKISOMpzH3DYID00oV6ib1ZQk6SUoHGp8aTZCHuXr9LFhQhnWTnIeMw4g4KFJr87sncemJL1ku5  
+batOPJeUIfCg9GGFToCk+6kvT7Ed44w+  
hbgMqJp847pSaP1UqI10YeyUR1jgRqdnsQdyNM0Ct8zXvdpXotL+4  
PeqdK9VDKa6xAeDwyqzN512wfZPH9FnX7L32Dr9gpEK5KrJRvLXyRUyy1GB0QnoDQ4srQXHuVpP8XpYq8FjT  
/  
gUi9PmacR3XY9w7pzQY3RF0JafVLm2f9uxIVepaa9bShzF3zj6ctdYrao8AuvGMLcsI5iuQQtaCNX  
+  
Y3TGik6DPIsnugfAkZFXpJkkz9yXpFwigaX8z50nIG4SzVv6oboZ8lfnY9Vv8r6AH0fSe4nA7Fk16AcKzECp  
+a9bNTNX+fHXXsIhUII5phFSgSFp1FYcMwsTU16zMqGcuJGSh+  
falEc2xSF9vZ60t1pAkSEhLRBX+z1X1395DHsNghP1KvcwufQkrX300ddh7PcjQdSV0+  
srllfX/6Y9wAoUufV+9a5C9ugcM69iEWKmj4354TKBZ0jSfo/7  
N3pVuJMEAbgt6qzkSD7jiK7gmLd/919n8sQggkmohi6+/  
kx58zozJwDvvSS7qq4WmNdUjW7QGrS3alkCJ67QwD8CtjXeFdyV3DTnW1R58/  
uQPjEPma7rWkPvr8hRHXJ4GwipKKZZFC+  
M31s4xX1hqF8ULVibXMPd02hs7nzMH4TQQdF0u2N9sbwq1kewZnVoFQkqESjlq79axar8i7sKPWHor1XJ9S  
+z08eQHccpcJ49rSpcxX5uVjYnH/ij0H4ZUvRUd206Yxn0WkL0vtKmao5E16C8ue8r1EZhJ  
+ipmD8MnL12417VRvDbPzljZFK9hWKRd7ay0HNmdy4yMvVqVjhzwmY8NsI+  
nRuSjQNTjqn3hhJxnezDzoBYBD24sdsJwUbqferNuclPhMXB13XI0phFeZi3ATZGY+  
vlxIAyGLR0U1Wd61yRN2vOU8gI4sP/6AQjAtoiKaMTro7S11PN0er7SD+  
Qm03BsCA04zgrr1JIuv9PDWenMk4sjk/  
j2pdJuhVOVVobGN1xloSlEwWkee5rotePf7DNdjF8r7iVIYNP1SN8WHS3d3Xg/  
qKm01ztasYOAm+6Krmmbn1zsf1m5U0GB+8puz5c2AXvH2Hev3Fb89AiT2508KqfX5+  
LQ06nkdmjE46fTpjquaILUXFSV8nv10pIQh73HKyB3X1V9og5EMY2/  
NwqQlWlKkNelKf1F0Tk46JJFT6KRujMjhLMGK91WSpzTzkRfZ8ewa1BuEyFqIXA5N06Krkvh2JBHcqJ1T6Lg  
+CfnZ++  
eZnIBcit6Xgx9M0wxLU1FH7U6Tvuy7p3YgxAJeuiKHwhsQCnBvbfnd2dKqNi9HzXkusY1jSh3JIYr00RauC4w  
+mEx+NcKQnmLg5wno7kHcYRYgIzEYpp2ZxnCkG4oEj3d0LNzEl6+kS50+bTe+

CqOmKGFUfJ3kMVhPMNbc5LM9ekiWhODc1J+k5Uam+LMf4Z38ux8DkC4N00Qtm7ww/  
 o2pifMMRleqbqV0sWZG5JOQjvMaELZiPCu7afc/yFmpsSevdqcX0/3RutDGmcb48JutP/  
 crdwYknRgIBk1J+6eXLzqHn9D5V/3ACI3lNjq3KS3tR9BzuKAcGEN7ZMeLMxIOqFdy+xNN  
 +0B4E+fBF3EFodfvMVZxnW7QD+1DsKlNXRqt5iqVjWjWRNjUZcsQSPC0JSkGQ7wSGJ+  
 C4Rv8zo25+U7y/  
 WkafG4WNOQtmyM6FZlDuqjfnIfXmmWcWjuSGxOxtY7240v5dp0f0PwNvqWoHg3IjNm7wwMQ8miapJwd1waex/  
 /YYI/wMDNg+bvzAZmJJ0Y41X2oJ78bReEp  
 03BV1/wXTub86+  
 s8ReIgbXmu6RrGIKBVignxEn2cMSdSCxuoVgI2YMy0czwNxjoP4j0b5BjzMEZYrRf8iQ9pV5z0  
 /4  
 R8GcevoGw1H5z96rHHQYPNH6H1PhLMzbkADB0F8hpSjZ88qJrrQ9CYYSF51PDK16jv20gq/  
 H2u5JwYVDSCTP6xjkinoh/5hrdsxVfy1Ze5hgRei/6Tt+V3EfGJBOMd+  
 ZLNiWD9Mff8V96BqEohq0Ql8sIhL/  
 D4JnOD9peXFPW6QADy45ku0v71Kuqo06MBTGmYuVxB8LlJad8+  
 g7qMjDkldsbRjTLLs4e7giMI64je8ESVPh/3ImVSWEv0QEt1XXN+  
 otGISBpwfJNAUYCXx4Gtrvgor+f12Nt3N/WET4Wwz01oGuUXeeDBrSAUL1RG/  
 UhzaIMiuLPXgohuyNtQK6+  
 HMEtJ9D0ThrypiBbB8IeLqXLME0IOxRSw5sURC7dsM9v2kJfg4JQHvXFBG1W9aV3EcIeIUvajzJ  
 /MxWyZsrXVEV+WcCQhGuzq38DlzxPdX0qI+7mODHrN8a84jtHYHm2ZfD6wf3+  
 qJNIPug31VBxXI+ECs/VZlUoATAbT9uQ/mf0irt0xhnPMgc10/a+  
 ECNivPtkvhsQAGTn9UoxZC+  
 z3pvuR4F8kb9T4d1u5rD0DwPs946f4Z33JQDHRtB6NuNuGieSnd6y+v1h4NRzZE9dd1DvL+  
 AecabzEH9EUwARUpiQbGSmmRPvhl/PkDjtTsVXvt5XA2XTVrvohc9xh/58E8/  
 OhnvFdBHwCjIwdq7SJBZ7sRV/KeTAURE1513Gs/NZ63nXqc+  
 OvKu5Jt2WZNF9FbZbxLzqjPaMq3Z+4utG8P8AvC0s3d07DrRdVxe94ar04DdXWBb5X3w/  
 QX8TDrSKxyNxIreEeV8WhruBenruvw1mvio/  
 GyMbltxoEvfd7VDYzUy6r0mkyqX6ReHKGN8fXmY1pvx+XB7Hre/4  
 Efr058eVXuo3VBBCNxS8nXw16RoFNHrG9werhizK7r9bq7aScscc5F0tf4afoT2vd50l0yvZr6  
 +7  
 ZX0aQfIwa3152MSvt4dX1NC6SfxGslp4UFLgSwAQ1AfkulJKfjfgKRN580Szm2L2EmQr8uJ3XABf4xu0D  
 /LjXQYUhPuFmXL+wVQ5fPBPANE5gRciL7QL9+3xXnyF9j7urks3xmhq+  
 yvkw1qFkCQp0nyV7leUcaqLdkA4iAMNyjevPMBXD3SpJVRkWmbgPS/  
 bhfWUCHYd0EAHjrV0irJu6TAeIOQ0khVoXyXm7LtYZ1FS/If0VMeBNy30Myjd1mQ6A4b0o+  
 WQLF7nxS5nPSlWdf3xVx+MKYIK3Kc18r6Pry5wDMRqBHLmHi/zWYp1ppeeQ/  
 hH1XqcsA8H04KTDRfXojfAXRTbi5uWZm12typ0+  
 SS9T9yfVNznpTEg0aV07Igv0atN03M93r/MPIDFoWo75e6jv52keLhYHrfFv4SK/  
 qc35D1BrjYd0gAnLcuzY3mr90v/H3r12pRHEYAB+kwEXkYtcFCheKhURRfP//11  
 ra09tRdity2Znkuf4lS9zNmZ2J8nsxcCmJb+E0S0/mbgytBK//  
 JPB9Uj9q2mv8z7E6A7kxffFoZ01trhSjJE2JpzV4WHJ0im/  
 JeXAwDQT6RUaE8dL37j75j0nxnkdKqXjHABQImI8DpbFdpBDj/  
 PSZN3Un0AGrmrwvNykvs57MzjAyG+S8vXalbtM/uZ+Jsz009aPR6mv814MEPJrjj2h12mZ/  
 n3ej061/Bc/y9Rz43H+  
 yvurCpRn6Q8FT7Bd8JAmTguSULjln6uYTTVP44d33qk59f0kriShbRLYV9xDbojLjn5f6jluFm4L1uQ1i1t9/  
 +c/9P8Ywq8c17Xk39j6dpmsIaxplypF97Rvcv7qoaCwMpHaze3Zze7N2DGHmYH0aQZx03/  
 BIedSM9M/Dd8/  
 NoIO5QziykddbUG6Jp5Zt3r3FXFGRTI9MQFrqRnnwX0afR3Et1DqhtZN4RYRFEUSv5LqJPq003YbKCGKiPe3/  
 +EtakAdLR7wnl6Jn1YHbhrARd2j9cxhynomeG9+8  
 b0UdP0KvgKGXdoVGyFbX1FLnxViJO7QgTzBF7zg9yIO/  
 pL9H0PPS1yp8sXWTN2kOMbn3lP40Q39YrwnZCKYcLUXNWFDVQk6MqbgKBfNa1nlPCvD3pIMieDNLNYJMrb05/  
 /5sqQLY3gqah7h/md0xH3IXw/hdBZiZaVrU003Rh3fRWNyG1Z4/0  
 ruwMMoN7RTjyo7UqTWAMYSZa2qZqjndi706tUpAre1+IaCo6gtzB/d5Y+biJSmVz/PbYMXH  
 +Q+gMREnDWIWSH63VxZ+mdBrKtY1IZ9wWysZe814+  
 xr0frVwsdwKA0NxIkP7cxIAExo3oCPbOM7di+FVrFXtN6Z21hF8d6oT/Etek0+  
 azF0b30ZSG8eRHa5XLHqnZXUmQF+ORGMURGi2/  
 YxJS1QE0YN5hGZfX0Wy9c0xhm9MSCDQZBFTdxZjKEp6Ju68241xJU5b9ghCUZN10wziKge5kgJ8JGdZCHMfC/  
 /kh5h0iRnf2qIjs1aM+C/  
 GWlwnNGYAItfCaZAgL38bRIMxDJKbV82UhzDx4td6CMMCKc44a0iQF1FF0mEp0hrmeoP/  
 wngQVxN3A0evPolyrLHa5j3IFIYxbjNxdXFFY0TAGLXkjXAaUaQrHeaGYLm3hbEUVx/  
 rJhh7Meb9dxe6xYIwFhOXMIux8G6WWlkegbEH4+

hdrLQ68aR0tdb0b7GsOU9pfBpxHxifgLETgR/  
knUE09fKMqyAqvsAo9kFx9X09v8XlTraYRhPpoEvRMbKa0skTev30R+DdB+  
iyTYjmNV0vv4yj+  
Wb5yttTU9bugnd9Vml88D00JaUznkVH12Sggz2h11K2A0361BVkq3Usm3dCpyUqBhZTOuPe2lPr6cNyWMZ57  
+NfRfMKytgEyc1T+  
ic1PaHXVXjaGumEk4F87DqWlA5GX3Ly1063JKdsCtCWqdzPsk0I5pJgte9xYRLJCpWIL8Qjvb7GW0pn  
+Kvs1IsopY9FxSqaJSoJYeFhXmvtLojwFm1ktzCLJaUTzpVSURk5kfzgCb3ewgYgvPEk+/  
SjyVeMr0EOjKNZou/s3QlyGjEQBdCv0cCAwUsMGMCE7GDHGyR9/9ulSD1Vz1Jhwi7Eb/13  
BNVourV0610ErHSGfvBOR69n+  
mfbqp6whHSETm70WsOewk3CuiHLz51EPBDuPS1BA2561WvaIboHQQcqW6dQn00HwZja6N/  
SxPSKxxZDnVZdekLBXQSVfW0e7vuWyuRp7Yl5LqW9YlmiNqpTvhP2FBhbjn0ByxJzzv4rIDuU39Fk7sDd9Ey  
+/8  
I30FNC9mvNMdIRbyyGuicboTYZzE6d6A57cPVcrs6NSJvpEAd2t5phnoiPMomVQn5cx07tjE6  
/6jlQf8b01Ver65g20FdD9qj8yfcRhEC2D/  
oRpKpBVUTmLY2QRHTE2E5W2tDfSNHesuSNaowPIE9KbKdco7SPcK6A7xrTrvhEWls0z/5A+  
VDMlZ3T7TIPaMjgi+x/u4aMCumNEVS0vwtPa06WZ9rq6LONZM6KLVO1DetGZj87WhLL18+  
uQH125+oxvpNTMXV7Ee8LPd1jBDtRpppULxXPP+  
ozHRnnOgXpMd4V3F1SI7lqPMkzNavuvyN+  
q20dXnWVcI1yiA6iyNDY79XZCt1RAD420H1qW1xziV87BaqN6MHGM7xQ9Z0gf+43  
oFyaeUW66b9zW116zJB2tA/1  
xyha674Xq0qK1pe24bTp9E89YU3dgGi2905nT5H1i4hvpZhXQPeUI6V6347QV5x3n8drGcbTOruDSzMq5zgs  
+ZQhpH8Y0v4YD+  
yXKS1oN25jZelFqp65rQR8M3GP6DXV3w2uLL1neBM6uitTgIY2Fw14zJC7nzDW9f7bmY1/  
X083/Gp2benxbmr8XYB6xRWB6NnkP5wnD+nRvnXgStAhehl4mx4GjI5sB2U+  
eLFFwMKkBD3aRTqAywwh/Z54vP5U4cikBMSL9IB1bcmNB/  
Bkqj33QhCfpKN7ZcnVS0chPeDJpAynvBM94NxSi7aGHxX04FopiHeXAkYnltwHP1XpAst17qWIC96QDtxYtN  
/IJ4wF4LmChzLwhx7h6wU06+  
tOqFa0Whzt2XjW213F2Zu5g1K96Zju7YkptfWIOAx9qkIMSFLRXWlp6P0zPK3Eszp53nCFj1bCv17srcxazP  
+AVM1LkX5gvvRM+Tu6/AT7dlynPC2/U5T+5+D36qUC0Q620LG50HS+56  
CHYBK2XuxXngnecVLi29Y7Cr8GxSGuL2ExUmGe7MfAW7Cicmxf1EvEjvXl1tr6jPzImCqR9ELRFzZkiUHjSNw  
/budDuNGIYC8JUNDAwUBkITyl62hCzV+79  
dt9vT0nQ7Q354ei19b4APGlu2LAds1Bm0ou2bItgPNbljB8wECy+  
Ls4n3CZKAuSbX4y60C5j44ZpNS7Bqojgu6g7MAubqTBrtedfuhwaK42asC6CvJPjhmlGtB9q107DR5EbMxXG  
+uWcVbBtVAVtU5g02gK06kyJ1kt7EAdsEvPxwzbB72ucWBbelJjfnPWAT7LvqjCK+  
qxow1uRavEm6H65ZRvw2UxM32K1ewMp7TljW4m1Xfs0NNj9gAxBG6qyKD7SBLugMNBkVa5Iu0Hj9q2G8d9IR  
/2oa8dK9ocXoI+  
eHUdD3FN0w4s04QDDU5Er0JD3gvafohhVPxIHexGo0KudV1YC10rtK3m5SQMBL10QqMPIU3Tbq9  
/OFd4Umt2RM0gUDT9Htisy12011L+62  
wcdTd0P0YFa3qt0T9IBndWZFPfEeo193T8N4ku7PotsWiQvjAAhuPUmvQ9BfqB0Let09qSS9ZAz0J7  
+  
LbtK99V4cELDU9PheZgr4qM6sqGfuCf2aflKmy939FVXjXqhT9KbeHjnzfr3bY3V2FTvyGb2ZJH3ENmiCd6U  
/Y4h6BqaXLvwCVgps4w7nKZbwY9Te4EN14uY1nBfBm9ySR9CiqCWy+XsWxD+3  
pDw016D1QE07/RytKyG5W7NLHPRNYhMuBFnWEfMwh0YNDs1MhKZvx1Rdu6vK+0/  
CSg10S2oNKeqx+v2TX0IMwbemlow/SJFNx4dxnLtlms3AWTqKmluAL9fU+dWT3uu+  
gXi5Ymdwsevhdn2yiDw7UvArpak80uM4Ktp+iGHT0Z0JtoZMx1v7ftnZ4Ni5NMA12a606+4  
glOwd734gwbMnd0f+3Gd+  
08Ls79xTyTCR0I6GlqRD2fAyp1ZkX25jIXAWNNjucVWsFz9ntxUWPuP/  
Gtyj3NP7X0Pzm5Gc/wyVJz9iPEs/+cGV+5Q/AUNbUpS6ALFuNsQyDqF+V0+7Gq1n4V9w9a+  
azcgRCiplawjF/m/eJW66dBpx0ECJ2D967/TZ1JtUxjzSdim2T8B0/znm+j6mqyaAdcDHL+  
pL3NJ5YJqQ7Bo6YWWfrGBUw0S/  
F8EwSv9b1iINM6968EHZS5Cc2SaJ1jih7XfcHv2v5QxStDmn9pLbLQ5B5ZhrB9zDDQ533Bn91N1f1wzmlCBW  
/9kbqT46jbq8oevl9365Vkd+i+ivBV0sxt+OueJfdXnTtYq/B7jTP7tdfY8PxJ/2vS2Zih+  
JbKdhlN7GtEPBP8hOA9B/OQ6N1c9lcXLS4aGqRY1UkqLJL0WcIqEkkCIB+  
NS9yG4UaysDwF72GtDW1WFEE0nDKLtAvZ8PXBpvh3M1tIGW0i/  
uZONTUzhYjGPI7c0r1Ibi0hADgbtttqR1FT1Vx3wi0mtqUI9D7GR43TSEA3hTrh6WdeT2XH1KvTDS1kmEY82  
/v//12nZ3rMtBDJ0mrRM9N+Sb4E9CLQsRpxLC+  
j5iAA3pz15jLuQQHsQF4505IKHdMhFJkRMEeOBGaTm0ICwP7GfYJdhzEt1NkEHVP7AC4kx9m1L7LvQTfo1hfl  
/USRPPs7gHb6dkspfzEMejqyfzaCwkpV20A7Dt0qLCF/g12FCwPb9BWuisByQaZyiue/  
bmPObhxCeFFsKyklgkJVjRMv9N1K0s2uxMTdCynH3QBS5aDy/beS2tqXzbq6z/V4rde//vL

+IF+2wynqHFekH9oCjgSa8dWqIu1dpwSeXlXqNCSwXpb8c25h7Vxekn+9YjpnLrNKeK9/  
 y5d4Q96MJxsZrfV8/bx6fLVaffT6fLq8TdPv/zodJ03wDnnsETHFP3TWph0oxsV/  
 aBYT1KPz7fn7ela70eHMgVT7Han/iZ13aTVLoAHJ8tp9LPFC+/3  
 UagQMfuwm1eFhFdCmuAdcwA5xAS6UnlPbhkG3e4TtsnQG4W42Y3Ea7qFEMxjak7esKMeHGuV3pAfbXdb1hUV  
 +jKZlgAgZYx3e+  
 KEjq3pB73kuPYrs3vdDx16oVBtFwAx413t8X9oN7h70IyRH9tvUlMvhy4rz0oY87coUaFTuSAedAFLhiu  
 /  
 r08mSlaIzrOucDIC8tM2JCUa3cob4jeK53oZ1YBlyfSs0W3YFU40Medv0mWHAr11G3chKCwH  
 +Yb4F105nD/N8m8JbMk851Eg9U419RUzAj4yDPQdah6K4rB/2  
 rjp2BVM1DsJwsQV3VJX8qNxU35BV/  
 X9IiLLq15lDjr2nHofQdrRcatW9Eu7XxgGPRVwT2kz2s30fTt2dYIooGH3FfWgJwzryzgoAyDS8uVxfceOfR  
 +bjK7pP1TRJzHtSw0/  
 mlceUZuVFzdyFJ0mr3TqFtGf39zbR1GHbvyWEtDb9CUkoIcEgkRX1GqzS/  
 w4wdEztsifCMrlWHIXKzcd8byuZx0KeRHfpaLw8hcF6IExW/FbBXEOBeUk/  
 Qkkn8QDfWuA469TWwI3Zj7kMhoCtSVSQH+9ihGysU0pXTXt4k4IjM6Aq6IraY0IhjP6hLx+  
 iUpn38moAXZtPoagStOXxmTmLOLUhm6JSuid+20a+  
 gqz14ImCk0YKcdtcQCrLFPxT1iRjMnWCn4IuAZR5H30louYeD4EEcfwq/  
 HJp9n8AbYWdcc5jSbi/BaPVN2qNdLayKPa8leCPgpQ9vgMFYOG7Yz7RvnHxkFHTdgD/  
 CynDhHnZzCUip0Kkb7TvHavNaLaA7U19B5CsShaMYyksEjwyl10hG+8  
 lvAmhkkAC0AcgF0kZu+D+  
 Lla8Ry50A14VMzSVN6QbS1CSZ3RqAgkQVpyZvLorfbDwNnS2BwVwglcwZLth5r7sUy2JHP5tbp0CXyM0lRek  
 +JwUTBuWjh16wBPpWUJ6vuSQUIqu6As8WGRr6EKfWbGo0ujQoSN++A4u6EwpP3i+y+ZLLpfe  
 /gZVyic4NStJBr3LkYOi9vkddivMRwCJQbSniKd+jS4ED6/  
 lUaw6dwCr6VGRq6kG4o4XFcyHxQkb5/Sxaz6Gv/r03m5WXikYp2JW1WKvamlTQSw6ewBt/  
 SD6atakC6nQRI7NwG/Uj6Bo7CXxin8F0BvhOUGrqRbichqtOG/YXODWQQdMTa+  
 yWW5kdV7iD6LsgTVf0591bIO+gKZ/479  
 IVGUy8Q2VU5bdt6TnppXPhBp9Chw3s0VkJk1wFd0h9JB/0YetC/  
 dugCfBLwh0a8P6zYWeTYSi/qPQcfDMQHvzmXABeFxxhRE3ws6NE4/xKdf1TuvC+  
 hSglgpNjddZFmxwX+JQQ/L1F/0hZQAls5WHEBkWTrDf41BD8ra2yp3IQHEdK0xBp2m/2  
 wc7lfGgz+  
 R6WSQc90YL7aZRoxBJyrZxqCzCfrYz1YCKQG0sxwRY9DJEheXjVtdY9Dv6NFHzqUAEJhoxBh00qYx6J  
 /Z0xPlxHEggHZLvk8um8PcxkAISf//3y2bHQ9kSYBBtkZy/KZqaqZCKonRS+  
 todTdFdGcjX3SGYMOdtTtXiQctvdEx2b0Wvk610zx1CPnWJqBVdedbtGr0hu+7S0w4RINxx+  
 qAVXXXCVvSGiL4zQSYIsI/pF63oyjNsRW+I6Cu5z9Z6c6guImipmj7dohVdG+  
 IUEKThj1yqjxm0VE2f0wla0bVBZquytOD0iVZ05cldusHPEl3n++  
 i0t0I9LBzQFa3oqpPaJA+jrTBTfWnJT9ZKZlT7Y+  
 pCS9VkhT2jrTCjPrwn5claY48ksIOWqskiusWPIugal3u0ZVSW8Tvd0d9G8FV11gpju80PKPWssuoStOL  
 /jydGcaAltVeN3b715POx0bWfuTt2dMTC4rXkruur407rPD+nUom/  
 vtaLegG6mK4ckBoQCWqrGH9B9F0i9hr51wjf8+  
 G6qToOXORHYcGmUmsvhAC1V408kvoX2s9lbC0luslztw2Ee+BfCI8IHP7w/+rzOgJ7N+  
 X3NW9FVxypkip49Lfqa0wf2bPKyHW/CfvZZeARxUo+0  
 pMtq8xxhE5Way2MELZdolzjZsZ70kUHPiH6CSux4cNh21mE/vRAe/+  
 XHie4MAaEeGCTGQ5q3oqu010qQjvm06CNOJVfChzvJZvhZeHiGLCYNceVLQmKwNughWtFVxzyQPbXgz464gk  
 +FJ7/K/y8t38N88CEJkwGqbGuWMwoR4QAof+BkdoqRrzReLg9p4XPaIP+PV3y+mz8Lvp/  
 KOXKH/UWBH2vjbPgce7/RU60PJTRS8ix+VUck/4bv7kCYRu20v6PEcIXfor9KG1BtH1ET8/  
 Kq10s+/ND++7mefyW74TJ3erwQ1EVZdTGXxC/4D0yVX55p0uSI3oC/FhaaVh0jsui0Ep/  
 DXxHgCVfxZV4G1q9BzB90g2nE5soQWOFnOgMC4R2Qm8FL6zGk0HC8+  
 mTxT50wuEkV6ie2GNngODsUvyKC33Dh0LWk5oPHV/  
 Fx6YpfbQYmVhkmzti2k7Pv9FtlqJ7g3r9BwYLMXVPUH8oRfSvzhFbwgtv9A3oh8qHJmIWApvLd3S81kCgHq0  
 +mJNSRp8BKz64QiJ93sOTXN92tYIwSn0AaFGEKxIOHF3Viy3q+0hy++8  
 ksffDh8GcAJVvt6oMzJ3mvkWW0We49Eox76xYoA/oidT/Z4Dg7Ut4vh0u9+  
 kFpxg2WbufhV17e6y85pb0Fr+JVrmulde2AwZpBP6PW1fA6BQDHvVRHQJngODLac/4  
 uz4KglTCy4ZTr76dRAVq2QYYCu5BCyJ11SNMbaap+2ZeIIiacGH53XDYPLHGzZH09U6zKz/  
 XylEAGtExC9e6nZH47AM5K3k36Nj4QnjfBbaz+  
 fnafsbA2FCLQI6p1n9ni0Y8R8FcmdyTIZnx6/LV3Rc+oX3vnrtB6wn5I+  
 gYykpdpwhYpefZ7jzON1ABfR0qT3DqplA7CK/Oo5Lz2cs4TH389Y1fggD52yS0uy+  
 dclqvdfmxxiFQHFkg7oQ4CJA45217AFWQa3AhnVM3gxvIW6J/  
 fNzoztcPXxiOgsA3oeVvEMxI  
 GraJFU7bR/z3tL2HUAnZjFRH3HPxJXoZynl3vsN89thbC2QWW3CAVTYMG+55



ZyqsbsNC9W13WZ4j+PHtUB4t15nVCMmRncCPvxu8mJDZksmrSHRESGwqmfpanjUq7Tmw8gl  
/CZ900mZIDoAMTjDThA8Ya+  
j2oMwmi7NqREeA5cWOHaqDLdW06NI8Bwbzb6N5d9yYhTYyB0iPfrZrGK43G00z0IFNtD03SBqTskRHCBa  
/R10UQpUclRad0yAA0eB3S3S+zJti0SACWG8xv9x6cBfL8X+2  
N0z2Pqf7qHSnBSFxaSw4CuaeatFqud59PVthebsvSEAJA0DruDc3s1fzYZ18gwlprpX02dgSSVGr  
+q3fq+w6BI9BwZjg64pguYmfiD+  
i0vfw71DJ2jQaf990dVKdb84DoyG8AXNTXbfCXkufhfdeYXm4L95dB/7  
vZdCM9g80rTV6I6eGFTybklFoNI5sINAZnC5zoyemg2JbQAs2XEqeUD2rAE/  
eSJRd0FeqngxbX8DBhWDMFQ3B3biy/McYRhdlZttyBwWcXhw6Q+xi46v+  
SG7x00nNw0sKnEt6gOD6skjUhSZngODjkGX8MJshueYbT16Dm+5  
tvSVHbcSU93F3MTLaXtNZx9B19RkYgmIJrxEt/eN0Jz544VY5cruqm/  
qeexg3ksRUSYD1rw1bdex3o66ngOwwac8OKsBnjNrUxgfvoq5bh8S39LvcVjvdA81EuPsmEriq7b  
/TU+NEZBNfIlujxtwxGT1Rw6dNReUfffW1yOD2F/  
QbdrIjGN7g0qiLPCtuiZQKxVFfxdYIIueov0Br31ZVivtza4sF3Q9Gm0CnerPy7zTsnx6xFgydIkr7Q7HcZh  
+  
PnU2eBn41wiMkpByl5zIpynAegL4bzsCw2LzYNzQXld2YjHM9X09LTix7ausWgY1d4reao5fC9  
/Pmt5iY6BuDFE0+5wjppP7F6endTcEPj9Edy8Vd53q4LjNf5rmKcf7h9q0+  
FN7bTbdMbBfKiBv4N6W7hrC2P8J5pnM4t/  
q9XVvkPy7KuF7sc0vtrHhcSRTdeiqg7weL2HNcg+gh4V0TREgXpBQFkz98GLxpHs4R/  
H1hPB7MxV23p0nfVLiorXkgAWTkyyCcCML926jozqJ7wnPaMRDBnyiiGzdfQJANwlLvcI4Q7mRZfnbdekn3T  
/TApTsocYyOyD6V7vNPws+ng27sfRken/4  
csTkH6aXn8tG7pyliRLip43q0zVR8dPc33VU4Rr8S/jLCHw+T3ewsFkc5B+  
lT9rc8R8agRvAMuWYRxEcRRhziCb0bKmj6WGrZCYSSyoVfd7YvxWARuaHohjOpwpQp0Ql8hgulEWqHwd6lv4  
/ReE1QeCgJ+okpuo2jyo107T1HxC/VZpafpfkw3Lyuk2Q87nR6J1YfbI/  
HVWfdZ5U2mZFPzFQL6daCpGfK8Bz4uPDCIOQxKcFBR8+/jtxoZf3wNRn3Vsfr8jB9H+  
wWszjyHNT2DX6tpB3vq71kK5+RapP3gJM0XIErLYzhoxEeBPEnpAL/  
sHee26njQACekdxtTAnN9BpCnfd/u4VsC0yBUCwjzJL97jn3/  
uMC9oe10RR7PP9pVe4mb7NGZTWy9TqLYTnyQm3kGe5EEtV0XtkloBb160Z6TM9zSt6LL5tR3g5uDQfjZM2e  
+3I4HyNv+fnUXmvWZpk283oc0leTgM/lJeP6uc0o+IxuvE+QXNeontD0kcbMF+  
ZSuhEYbroDbKSBwdeGbxFqJBhJC15z0JBAQB0i0svG7a9luS0vfIr80KpPd0NxlpsX6AnpQ1ZyiS6g5pxqVoa  
/LHPBcAzfHucmEuqTCE4jU03uLT53XCprF0jWQJsIglqhOUF6vtpPsi2xFaET3H3/  
IcAd7mkaSX01V7h15EeTHee+EVuD3ShxSiuMCK9NPq/L3RhANYp00I3So9x1/  
zfODTqykPxlKl0Hk+heSHcxXrvmlUlKSfKKE2HU7wytt4JtF72n1j1/zX03I+  
nlTMD2pgIpp6W7zsM1WuQNui+D75YydM32IF2MKiVxjBKpgvDUUCtf8hzArZMGMSUJeuazG  
/OSH9PFijQyyvnB0Wu2Ju/zsrhx0zl+NFxtZx6oI6BGj/  
PXPNeUM7210dWd3SbdDUkjFUDIDXrJdLZd9Z3btvttt4j+kA6ZQVKPe/  
0xjLQwUPypZtD4j9EmfUIakSVQxo2brcqoGtKBq7K3hcVhdws81zVsdm11TcsnZTZLd9EhjUhEKATXS6Yfg3  
/53kzJD3M1URvk3mCKRfTY3oEHnPX4Eq142TtiY6KHP9tQhGsHozG/  
TXP44A0Ube7puVAmDFZu+0YdFIHhIIRrpeUxvP0ui1qC/  
ATUEbAxILqkXQy8ccuki77dNS0H5ICL6AHPZAAIrwD3urdqq/6  
nmotiwu4V0o6kDnfPcUHainRqWhbEgB0TORuklQbkbBPGhh7s7zdr17+  
ENth3DWuC51vVwCAoIYNGbf8hkjz4krSQIufDEw52yUWBhA410s0iAORvSiAQF0Hzx2oEOYWF6i7QS5vvQAu  
+rQn7NQS0AZCW3M0rtplcE/cZCJJK4nlNS17nBqHTfob6WWUs0a14dMXQS+  
LBcI9bEyCHTa5e16SpJc3pa7u5mtaDozM679gU5Z7p0uc9Jlw3MR7qM+TFQj/an50+  
I205A0s1YafW2+puVAncFllLZFmmjmv2e5Si/bspuzWtXwuvzG4IW7S1S/Oh8qCrGG+  
poVH2F3/+YMDWEz8W37+  
Vd2UrsluZXZw2uLtOUI3IO3U1PIgGNS07ElN78jw4TN087G4dhrKuia7U9+2  
YhdegoCRtmhcxHFQ13+vQUT62ShNd2dQ07LHXxu+tAgRaaYNUkq859k9+eV6ZnsNkbjgvX  
/nl9jpHL1BjxENx12FzCgh+Cw5RIC9ky3nUvZpaSvidT/kd2q3Dh/  
Asw7TTQjMsEc0N6BTDwGMyEkDj0Ih2PR47CA0vviJ9nLo8Y0FoVuS/ukA7/B3f0nu6+  
Zr4HiUBxmPttdQJ1046uaJz5lF+  
tBNQrpenxuMJnGHhTFjjQZqW9R3j6oIpDxJpH8doxMmUM1JzkdJqlpS47AoDXaF/KLv+  
VfdFam3Y3ToW7551DhogUR0dRvLanrJjYbN+4mjGcob6KT7LlMHlunuv+rgXnMD+  
hkTXmnrsDMoYPaM/h8U9EBjvBCqhJegIG/fivruJhWlsuHDrDqX/9  
L5aMa5Fj5p53d2Sw77WCItgnHoQf5jbpAiLSjiw6Ki7E4RUrPn0jxmqAQCGkd0Ev5p2158ZDWoiQD4SYRZWq  
+UVvsqhSNVvWIjAl/YywwMORYN2T9E3QtuDaV8o4u4pBzdMdxIJMMuW87WJ/  
vmbmgU6Ngj8EtNr+  
af8YrlqXmqunzAT0QpbG6x0e6QTWNBzC0ggzHlWqexbwHHaH3CXFxWrefU9PmsvdBEBg4fe5N6TXORP8RX9L  
/oK52sWzqqJRJGrdrC2PGouiaovAShe8VXN6WV0X06eH5c05sgUjkZ5VKnuivpPX2hLy1M/

WRU14sHorH52cp60Y3jNNkjA072Khcfec/ODCreAVj3NfkpxFqAfAZOQH0TFj/  
Plqn09d06a+4  
Pppeb8u8z02TeCP0ZbmGQDaPPktWN0lCHRd6QfSULRmr8tg5Pmc1lS2ZzfP429CEb9knYEjJKAbzcZtHny2h  
/pvkwi6+MdbBgkx6WrPACEHZklA4gwzCqFQfpaCQUJ+  
mjCM0bVXnSPKh1f9Kcd9hZyJVYgliQUaqAtlep7lkoeG7d+WJSQIJMJzxpLg8x0A09n90+  
EXH70buNfa03fqQPahuV6p0BV3vSt6+  
Teqqb82Tl08nzXkvAz3DepEuqgCUGlCLKhfGCdIQqsSHVH5FB8MpkgpFQW7XjJjrTfDHZ4B5M04UljcEWjB  
+jKxSkul1yqVPcE+uevCchC0o+kiZLmUDkPtacfMegCoRQU+00MrAjDHYt6yCwOiLxJ+  
lyqVPc42cMfw/r64kRlcz4bypMo4TYBjSDOC/V8xbr+nFsnVZlb9CmbKtU92wc/  
hgUJIHcYepAPgdDtOCdPaNSFe/D9aZTUicEejJ+  
uEqm0vkr10bYDv2DTJWkj8q7a3VFIJzpvD78Qw8ZxkoYJ2ANCbDyiFaP1VaqPZsz8grWYpFLvFs2oBP9tQv  
/9jDpCLqxru7HzIAfTxUVKWTqaCADQ6bqAtqRCkB0wCgEbMk3J0rK7/1  
KNQavqAsakgN4UEXd5/cVkfzCDPQh3YTACLQ07QAZTjdb2V6nuKU/1  
io5mMv1lvGWY0xmlbB1zfW+BLhBSVvMrdIGQGI/FUQPQ+  
ipV7TMcEJIhmaAfQ068j1VA13HqtXs13az6SbXtSZRhFLnOAK2vUiXyJw9+  
DPb9Uu4kgwmFls6QVHY+  
XcefVxL88cVZHbDNpBAMARUyTgXQePaD0s5jw1qYF1ffF30GeTkG2ZvbeijpKtGqEQsBrwMhCUiVateOCvQz  
/xZdUs8raoIqrtvlUF513ckRttEWwg3X9nnOIV0md3dI5FS0/lg07C/  
YomdQDPg1Gn2ZXsguKX1dYBPh7R/yzrQ5bRgIw7tafIA5whEI9000Evb//7  
tiqB0aEtuoWFq5z/RDZ9rplLoPK8nadwP+F2jonucytuj8pPlUPS1Dk2Mo5+  
wtx5sOK9070qfU+XxuUvsme8+DvxEy2mbno0cytui8AHSpeSst5NIYCG82nlxqw5b+  
Kh76nUMt+IqQe4IiN+kLKv0ItevUJax+3zRF7wtqRz9ygBy4ff+BuAMxhcg+WPZ+  
y04PBYq00KiwnJOHRRJcvRAooS1kAuh+lypxz4efKM+TG0AhoDrLvn1/  
rgUFz7dSUGNdWnOXF+6  
A1tNlThwAXb3rc0nbYDykgg4bJvmMRZHI7q3e3j1IwWL6RLBxOnMANRGwRdcMdlewkrAc+  
WJq9LL7q4UnR7yDYkGLDKQ+7yvaB+4OZUfJW/z2AN0JWBGRA4swbrMNNgY+  
IioFhaIgdGjs1h1AGLIE2qD0Xg1KakdnDjcOdIvexnmJUC3sWXiq0dHXiWs3LVF/5Dwl7+  
lq8XxLTrx05QBhBXx7cydPHA/3xGSkaLaAuVoZMZNXS0l+Ire0/pf/  
R5r97DrakGXsnKPRXe/HT3r7W8ZtuJzPhQGxqgYSMHe2p0e3TyIi1EyVu7cBOV+0/  
qRjJnR91U2DvHs/m6rE8bj9vbEN7Jz1nOErogzd+YaKPfb0U22r6Glt+grwHu4fU1uL+  
oPNo/  
DznI2S2l0sbd237q6bo8ZyijoseiuT0fP9fbF9S36fHxXt9Wn24vJodeeNsOgwsWVjZcXhReJlqsHcTG  
+1IoYagmCkqajHyFjfaoI/Tnnwv6Z0yJcgP64G7u9e3r9dJv4Tw4G/  
hURGsQ3UHX2I070WaW8hg5AgQbCcIdIpwtP/2qXeUhv5e7Xu43Vye3t6Pm62xT/  
SH6eMsP02trdxRb0L3ZCVu5cBeXQoBL7HYwIHRuP7rk0tz0bt1tHt6vXYq0u0jSSx4aw4PwsnPa8L  
+QoTlN0hEhS7sQpZQkMofL2otu/574KMuV0foliTJ164ZpzM1fu7s+  
PPEgp6Lqij2W1ozMfFJhhbe0TV/Vuy3SbTD+azX9j6DKCgmbuD+5e5usFEgY3JBAoh+/1  
JRCPfDBA8hbYMNqfrlu7MJ108DcobE6fR5PFbN159MEImH+g1aPTns0jmIlei+5  
qruUf9CIww4eBZ5cdyq1f06kaNue97duuyica3f64HvmeSaEwIs7Fk9ueRxIypL5ELOHuhLnkCc9GikDYANT  
+XvaNQXd9Frt/Fn00BtDNEJ67fA53ZCwNoMei0z4d/  
czczPiwvHsW0SeNn6bXGnDJjr4m9ZhnyDmo0N6YK2Fuw6XoJcidMDVPFWHP+ZFxfk/  
SdPU9d5k8sAFihbN5AbeRVNCZUJUgd4K42TVtmpYWH14r/  
mx3MD3cJKYjDKocMwINDM20eg038eUcueuL3hH0XWVycLJ/MCZ6dva+vukKFkmrjCUG2U/  
Uuampkgu6ruhLFoYZOW9p0hQUqvFperAHdZm6XFVgiyanQjwAt/  
FljTghVCUImGE0TASwIOyZ2DTP0T9/tP6U6WJIIcKqwjGvYI1lZg+64wh6h/5  
b9BIEzGSI7naPwj06cPvzL9Pxpq8t4ANaIOJW5q1ETMnfo2qIvWBhVI3eooHnQt6mZJyYXlkBoqqd  
/2gF1sARp0BdcJy1rIKuK/  
obC6NiRPRuwMbZ3ueDxW30ienJS5N3BHvsU8fMOX0j7ghK08UiH3VEFzF15hJ6+  
dkHpw9S7xr7iRemvyRpAmOwiAr5R9qun7iDJyuZhZkiDT9QXJIUk4m8Z3w3L/  
q0Dgj3NZ06NQH7YNyVeOE0vrT2Ti3RQViSVHqLpdtNCgtQRYVgdcAqdb408dLx0+5  
HfGldX3qio6Rbv0ZE71Y4P+JmTX2b604cgVXuIK8zjcB5SiK6J6j/Lv0+u0tb9BHg/cz60/  
SRB1b58VLCBtxHXNaajugI/n8punoyLnol/liFmE78C0bJrnrECw/  
cR0mbcEIRuh8ZZ0j0uvlPPfUBoSDTrS+Q8WqIR9CHMiBlcEMCeTqi1+  
WJbmBUS5c4DyKP4q6YvrP+rrp+LZ/2  
AUqBuAszoHT0b8W9PPhZdJdvL4cDQCj0dAV28f4aekPctr7QuA+  
RsMSGsjlEN1HRPf0zmEYPH0rf90dEqpltOwdM/I2V2zFxyu+6  
h6DcD4E9JZ8XLnrUYsNUtW8BbR7HAIDXTfd7Ukz/  
bgPxwx104mKUsL6vlpboA3miFz9lcc2mmWsOxIMPxPOHNQBkmm7ZqxcmvqTSKElBB3iRtUkFAZYh7Tmtojt7  
+aLuRdVhT++RSfyqI5QF+W6IuyiF54RfcmBJimj6DDLBNBVfKqzDL9VYEpst+

wBYPSPFHSIZCWid8oieuFZ/5Hxz/wK/5b+Q+211756j3mNwCL9yp/  
f1uVBVNgzc0MvUUnWt5UR0btMbJRgDajh+ZD4TC15UZ5heh/BCNm3DWvd8hROUO+  
S1u7kYU1EnWHCDzg6GH0k9+YzeTTtBqT8AeozLacJCizSuCzoJfJcWApsCFqir6QNaim+  
ontbNkt1o1XQz/oSb/10  
gRG8A8dYtsvffvoQlmiHLu007gBYiolMzAsolsj0R56D0nky1bPnB5X1ZDF2jJnqluVKCh  
+5  
PlLx03VJB1kdTdGF3e9juiq6y1v0y1BL9PZnM0xChulT23ZFbT4TbKBUovsH1kMfoBSiF17RsWNY9KbWDnOf  
+s2zVk+j1K0i2UysrKkrN2DwChFDMWUyq6m2F/tNQq6JPk91AaiF+  
mv9mPYRy30IaGoMAVUCnIZCNH9K2m6ENZMxaZqeihI36VjRKiTkH3mxwTR0m/  
DRTCGRFunRdL7cj62uImy73HB1/  
anem0wbT1EP2H2Zv0btHpCZTuSovS89oRGn0AJcHwEwjRgh95F/  
Glk9Py9cehVQ0y1iB1MT2e5JVfDcPydDQ8Ljnw9QfoEK8znt5025Ck0uyY0Nw7UNDP1jdm7dpJgG266P6Ehf  
/iN18GBLd67FJSPP2a49jWn5GBQ04HK1ACgiDkHkE0vlt+  
dM0ID7T7ANCCh9SNulD0FKK8ehcs0h+hU1S1UvbV100efUyH1/826rtFzkVdMa8E75yP1k+  
eGoFxF/sQLlwGkd+aUQvuqIPzD6yiV5/6rjJzMqf+Qp/aP/A/fPUoBqKfomOCgH670kt/6  
QHCCmsheXwD1Aa0Zm3KWq4NjKP6nq+dGsnOV/  
yjcggKaIfmbTkan6yPPqYB8TfyDpYqE9ZBA1t0T9YGMrbD4pD9dgb1D/  
Ivpj1uP7xd6ZdqfKxAA4mbCIYFUU30qu1dpq/v+/e1+  
tVG8rqJSRqDwf7tJ7zj2WzkNmysSQD4v8xxIiOEIyFBvSd5Vatb10KKr3mSsTcvaxSiy6tJyyzXtFNg28I9eB  
/X4ToY3iciB4j+  
n2eokf3U90u0T8VJKGGLWuNLP095XWH0B6jBkr6bhy58DgRnfSu0cd8Q2gKKTH7vGWTbLCCDW  
/pi4yiQsCvGbtBZ7dNMSlASEgVX2F60d9YHBsTfnKnS/  
SoXmfqc3TbPXu8tuVV9nlWjkSW83nKiXfovyLux/  
Sp9KVJRMGcdcCbSiDb0gIEWkz42Zn8102rGVuf01AdRbEl/  
EGS1TPj190TXioiH5a9Dus6H6oBZL6VuGreX6JTtyAg10ooGnwTQtUfXHfkLcgkLOS/  
E5GRkBHRCS6gX/h6oWe3GxqI+5wdfgBIAQAwYjdu9iB4nYYa0ir7hm0HvkB4Fr3xu+  
a1gTLx1VCzRf4CqsnKy7YzQsDlX6JAs8xjlnpkXHiDoAW2+  
AUS1zboFR6SvMMMfCb3tq9GUruAInBujVK06mlSV21twGoRMdFFcqOgmnaJ7rBkiZ9gOrGjDNz0K9gPL9uAO  
+vnCKUk6SlcjJr40MSv0r9wql0ZG0TusETLkC7+xz9BA+BsIE2cfZ1w4zRtFO/  
MFX6AVjvvMTKkzOKTuxv1sl/sIfZ05rk/0d9YDGXZ9H8hVjORp67ozV3twCt/  
gHZsioH9hhf4wsjwdCRfvED44T6run8fT0wtjGGoSHaHKmUOGU56v94FcYYYftlf+  
rp6s4u15MxVna3vLFxxpnpq3hJHXKHn+GK0MhbBL247TJzpSxpI79rC9dL8CuULIFAU1g/  
eMeurff2s0D+vKAivsNA+Sp4YSJ5Nuk/NjmcV4mrMs+rpEhxZnh2GXu+  
MKAgBGkmeMCApVEWgMGi7sQbc1omhUr54+  
zd3sBX0KNP8jFCQMvTYT50QNfvAQpd2rPztzikuxCvUvVW2qIZAnvYMX41kj7IXT2bh+  
GNULS+  
jt71vhfYxsZibtPQUwv904qkEWIag7YdM10sJrFpKX660lFye5NtNp94tdsmp58HyP1RpXs7H8fLtnhI  
+cPC9N4JjHudjyok30F/4b9suiHVgaJE+43GJwBFH0W8T8qT1Xuyk7ZatfDVDW+  
ZTR7YHKApyHkppFMpcrgBAhZi/00Sd58o5c7PCyx/  
DMnsMsJpjryr92u3xzjGENIs8z0bGVhC7RocEpMcqLm0m6dhR4qxKfw0m2AJ/  
Yc4SADc4cSqhh02Dim1Jq+ggKM4waQorZfqGpXjBCjVNApXrCxpt+  
FMBsXj4001tK3aWMbmu5gVDRoF1CV0uEzm1EJ97iVLufFQBUT4yS1R2nK1L9BFfi92fjys5SR6BCqA1WJSN  
+WBSd41XD+Nw4g/USSW68NNudECLNM00BPE  
bvbH9oEn1x3fMuN6NkmNwkP6g0Fb/92100m/  
ORP8NCc33Z50F8A6wq6ySSvDuqNf4ddg86d3cmWkRXYF/+vI3+67  
oiQfJDHdMt0YcR8aFyBmHG2UMJKUjWnIk1QEzfE8hBp4Haxh2CK2nf3QjORHRffB12fdQJ5Vi  
+5ZB/pyR9qHyZvrAGjPgikZ9MmhRnu9od+  
ZMQ9A47BQuWYay0BQQMNuiSW17qfExcAsPBJOlrou4j9hN24DCHI/  
Vx98fpZm0xNvZJH0TOSKkp1dIi0sLzA8s16ZuUQyp/6  
oCwFwmtB1L3YtYKd6XqcyvWt4i3ve32GoBeEis1yqIHS8T2+nXnwpblfMQFA5WDdk5+  
VpeRNh+jEs9jMsmFmltPw3Q9anoId6lZjToGkAzZNoq+  
SHrz9bTnkgHLdYoPtenw9ordN0I35ypSJ5vW3SXgI4wjnecjkOP8guv59CNr1E/  
sVK88JtLkYPnsCjBjRme0QTolZTCFoFfTgf1CX4sk92MQs0onXekQvx2g+HM/  
cfJfJWGM25kWPGBmixzaxVdDJIJyvZiZcd0nhQRfpxGM9ojsnp+yjSS/3zTB39/  
DLg6L623WmmVgHTVdbNab+h5mT5ZEGYk7Sht91PamF9FvzRW2qIPc7YLYjPe5DsUy/  
irauBJaWrnJSAzfn0abkVJ8gboMGOKMfmhujmZu/5YdbdUatWKWfQ3+  
mWlJfevzjq8UIVN7DTcGIhUA8Ah206B/N7beGKUJzwDrvaBaeX0dvyJooeXGL9D95Pst/  
vCnoyEl3X2nJl/mgI81Lfk+KVdjHXYj91FwVyCOBNYDcRB3r6XP6V1C/  
mDuTeSOG2ayOr7BJXFEyXf10GXavIVGRUC/tpcNE2sgoSSGwh/



apM9FvMgVyCnvrq0dKh5mXcZak0YAbd7xUmguZd0dmTjMf0JruCABBx2Wwa7LYuYgBMQ7Rp4kpXBKxU6cpMI  
+RL02nJ04xJIfq1BEysC2IfTw8jb8ipIE+  
E54CCtt3LLQ0vP3QNrvdkP0xvUNlRBBkXV1Wvw1wxst5sKyYg+  
pR0dCFvcoSZGNFLeuo9NyYyHvW/CRjfvMyEDIU7YWqzVvzYfj93LnpDTAs250PGckY/  
PeIjNt6zfN9yS00eDekIVv2+Re+JEZ2C5xjwv270GmMZg+Ee8IasmTVAhst06oEQXDGic+  
cpRI90/I4pf0BB7ltx08ixYvsgd093112U6DUpz0QrG0UoVQ0+sBHz5heNW2ftjDD2J+  
fbfC3vUga1GNJJ11KeiU4UDIi32GpaP8rDN8bFtfTzLJlY074V+7  
NTftUgYmYiMhzhMIjPUJaxSEcI5Yj+CY8PgmccctneXJToaEpMneM/9Da/Mt2DR8eKr9rrB2  
+h11V4HU8t0W+  
uFwYmQjApCCC05orfh8VHQ5B1VUKAA287R5brNFAR0lZzQDzE7NT0PeU6kPQJAo86JvIoI6QiB1HN04pWEJ6l  
/k7vXlQEMuEiW8CccJgRLUD8evPA093cKtFUGosdx9/largJXrP4/4  
LsHw5Ur0cmFBwmrDEN+TS+8MKzC6T9JAuquBz/fHDGZqD3Wh1LHXomYoD+  
yiUdFtQcAqryTclAJVJ20cSkPuIYEopPLG71fIM2+7hxvk30VIhhF3joDp9hlDwC/  
X0xLekZGI2bR9LXt6mo4K2nJk7lypPILoCCIZ9wJ9f6/PR/  
L1j5V9mTBoDJr4tq0v9DF84kb6bq+mIABMpW3FbjIdPdt/6iwrAw19vXPwsH8/  
fZiBwAEF9MvGNcRqAl12F5jP0w5xe3PuuMJ4vyXPmB092R2j7CvBkuzWFEL4ez9/  
bjSKoRyBYbSa+OQtQG0U2SoG6bYM/x0+DQK/iS2qPzkwPnuyuwGdeTQEQMGZOX+cDZd8tVI  
+yuuY5eH5xXZCKwed5nykA0NxbEbcohdFfvdak1m4K2oZ7hhxYBQ2biavLOH1RgRq/  
HM3fmwEUbFX7GDJxHgZPi47gXfbpnPf19PgkHiNSWx2hdiAcYYWVoPa2WpSJtwjaiPsf4gE8LghQ30  
/KY++fK4Tw/ShVjkZPf9SGCsy1wzlh1MBELQSA12S6sG1yf9cc3Yqfbf9C/  
KRAWd609RHU1oNvd1gyeAfJknyfGvfAySL/sXenzYkCQCGe2a4RRDvA1G8I179///  
dRja4JB4xaKIy/dR+2MpuZSupvNswMA0FziCCZxNHWDSV+7  
J3UuZw2yHjzOE70yCqzqXY9X2gn44juJ6r7WudqYDt2Lo0i/ynTEqbnvaqM7WrV6tHm+643  
DXDHZHI1FASTzbJD5pPcbfgr2Cg+9n3317owM9PsPoQP/  
hKbyi3z7wdmfhAoj90xt1UtInefy3iVD1Vq8Wh+ZOaBGaEbTqePwre3vr93XwehqsKsf+  
p6N0ilrb1bfxbEb3rdsdJEobzXb/5FoyGnm0ZtsDM007xz4Zlv10Er98w068C0z/  
U210TUypfZESvQ8bP8toB6eagx09MWiGeeIAfsTqlPlKtRAIlZwvALyw/zREFzuHh904+  
hMx8McKHE0L++pVVLsf/5  
f8ZgefIv4Xi2VbYfkTMyhw6B2j5mAnbZzNmDPTaCLUn2RP8b8nMB9FTT3NSnCcJ5HTOMQTvBjN  
+D86kDDBb1Un83LpuOfSglJUq/axwH3rNQ0hIdDdJl0d7g/  
jB8Mnee7uovoQvp12MAjTlmRmvgjEr/yqgP08jF0y8fk2KhhyW+  
kJ71a9S1bFiZEai74HYWNwad1qZvIdVeUm/133oCGEA1wEyzAbrCw/uCyrTaGwf2a1wZJj/  
iPMewmb/03Rz0QK06DfWz9Pfa66uhRrWXi6bC1hN7b01kpYt5Bcp/  
ynID3Z1Wt6FHsZeGwIkioQMMEsx4LeCqfN1FGZX2bLFzqPYyEFgHZehL+zDUx4w037/  
H3mufRE2Tan9xAscq/bi3cw84DjtU+nX4e+3  
LcaA94YPWhJ5f08moH35UtZjW5H5Su9GorejmuZflKfZSoo5/GERnh+8K/  
GK4brSo9dckqqAWo4sZcwlKP39eAI0Ijt5fkECVtkljadZrCzMrXZmrDncrfftc0xmT60JfgFLc9Tbx8cBR7  
+/t5K8HKfMu0mlR+uDSbwamXjWqNzfgbvjdJr+ggSW89iVMc6NTq3b9wR+R1uCKpih6/  
Jd40UxWNJZ+  
ssR2INy4VzXB5M4HJn4LSGEFST1tTJH723fGQb9MIniWqs6rRjGPnz24xfg0ER/OQLHZXm+  
g3HdqFRr3WY2xMWFwjXb20W9hlpr7gx6AnOE6QXzcbysDt6b1/  
m1nyRA8nqGZXgkveK2W4tsiAsUlWq3gmQ7ceX85ypdSufQzfZCfYcZYQXhprYPXmfwjZgG+  
isSJXgkXQ/  
F94lrtunvol5Hhz2mW0JZ6Ts8kkveHIXxZ0oa0pyhDxIkL0hgKZaiujZeKtwJVttJG1ROPCt9hKeJQ  
/BmkNSq7YrB4DPDnSU0zl+Tw0RZTtJ1DsXNHdwibNPAd/  
GqASmud0IfuOHgRTJ3bRgultXGtD1wXXfQnnYm29CiB9hel/McJ+  
nGelM5ZVDpcwOPbMd7Sw5DnBLPadgoXa49ZVue7/ueRbtLvTiBHXiotL/2MjBbwG/6  
LDULM83pYYjTZjJfTGSt1+Z0u80WgMD4USHIzI3qxeEh0y+IcRgORX4Hd6DET2JQx+  
vRfh01ERjwIIylw3wnEEWta0cMgMl3rS3M7CdTi59l7eHpM0giUdAUHKA084WPe5vinXemw0T  
/G9XRYQb100q/fuWiEIlqBNbgj8lhPu3Ntf8LPEn6gUIGl1+  
VQXPQx1r2dVkiKv0M1qTDcfU0DfhhbJkF7LJEvt//6  
euEi9QDRXJ56VbJAXLUP9d0MEZwGuft+DNymHfqfU1mjkHh3ds4rPCd2BqH/0  
PcUKDkL6n001yPSleMwNoftSCHudtLvNxrjct3vkC/6tnPfPchTaBWrdCpZ/UsKh01Qx1+  
BOMMWb9HQUILzlaneKd9+TnmAE7fBQa/  
cPnf5vQUd9pZlLpavmje2Z4esge982s8pS9Lt75zMaUNgH+6Q/02M6+0HPBaV/  
nU1qOgYRqVr8eAucAMKiH3v/4DrRW8QtrAyc7c0eQxwFmQf5CG+3rfEqNJrpiRbt+  
jazc6K2GAr8qfqMMMF22HAE/  
XgqoRAIlZ0uvZTnCAGIkKhG4gt8hT8xZazyy8YR64QB5dtfHLjeuWf5Cm4eSC0m1LKdK3yBRicAGg  
/uTt5tPuoGJJy2KXz+

HSC4lGpDDcr9pJ5jxe8Ao9aPSu7Qgp5aRLOTula9l5SdEUJSenWCaU2CHDxufhrreM1GyE52G  
+nHpCZWulsV9S2cfs3x8aTfWMYOCOKxt3BNilh/  
iVwCdh5xZeQEtM5pR6Uels5BKV4oM5k6V72Ncrnzdd9b60Uy94+2qGQw8xFV+  
cDMwtoeN5MwNXVI/Kl2fU+1KsQb3iYCNlbuiNWPjRXP9hp/04KPzDeSjDlAgvg0+  
lQ7VEWaaDSr96HvZp9KV4lWA32WUQ3UTWBp+o28A3LjgLjDk+Q+  
vMCWfZpPcCCXboM6PSjfoUTa1eAPgt0SeVl7phY4pCjwyV2TBfVSBNc3uif1HGeTtYUi/  
bMaDfRTpQdUulKsCXB2Q+R83R2aWoFtbQq+E8iZQk4L/3  
NOHJ67Ie7taDXupAq9fEUTWheAs0Lr62y2CUxN4HWCyg2dT0w50Hku6Y6GKbN1amrrSw3Rot2lZjCodMX46x  
+kzj70yfVJdyQj/+3  
OgUOCH4TvZlG7FqZEDAx0aQc4owN3Kp38J4IZAPtuF3QmG4dGPfQOISMv1nnxvQ1Nue6W3fc  
+hnOMcr4h+k6MERK1CC92IY1970tke3Lid2rJ0JaN/0  
nnUk3DD6IHLlfgjnOgVDq5uvVx60KNxv7txI6Qj9f9h51LVQqkLDNg2u++  
djs6pdPIjmt9cRdvlujMdu07AntZmrVrcnQe0htLfdy65PkpNDhOUy/DU0ZV0nsF90gfG+7  
kFRlEp++smFnTHK5VOHuKtAvfBYSzwMxF/7ZxD+  
FYBciXGae2dPNsW8hzqGn6y0u48FuhMgVyJ8zck5GY7A+6  
HQ9XEn0bXzh1MNETUinT8fi00TSTkRv/Yu9PexJEgDMbVvdsGg7ndfYQj3JPU//93  
u7HdxgZndwMdaZPU82lGjGIJ5Z3qw10980G4BzFKEKKWZJpM1x9vqvBOKfUfMSai1E009+  
Yc9XaEEkT8JNY8QgEhaEniD9R/xXoBo3rMiEG40y5s0L8tyVUaxZxno9AZ1H/  
n46hH2dT9FgDhXhuRh/BklwoKW9ceMRREh2don+  
Kj5EmXdlrBRDuFmxF5qX3nTNeqmItyP4Lwqs9y6r7a5/  
E20kdLuoupgsiP0AgRrxXH7cY05ZY2yBbcA8lFvagPonRDkWPt/PGYDxE/  
eImWxKijqBJ2I9DwCAObNlvg/1aYx+  
RZT6pEofjMfwSIwYqYwJNwhYehIzUnsBGPV5NmNQd2A0aqLUp9Rc9FM/JzmW07S0qHdDm/  
SwjUuJP0HdhbBpiVKfMB/Cx80mHUk0+6VJP9ds0mXhL9M/7n091XIn0kZy6lP2U/  
hwYCUmSa8Z1W+jThg8i9UKk3/ZjDTndyP4+pKc+s+2  
dTAEr2h4WcnuvJQlPXiTPC0VIDSDF7Mp9Z+MGAwngq2YbB6+80uSjpERlXmjC+6  
PX6vsacd39e/MCmA40hYjWVGf3L49Q0C3IkZEF9zdJb1d0aSrxf00QQQ3CL2a5NS6Zuk/18  
Q6kU7QHST93NSkq39WW4Ph0FKMWEZkNgAR8gjY2NsXn7RNpBs97S+1/  
lFn6DTnhL4UtDoA3yzJbe0ct6aacycI9TdR6kPbKRgu2a10T+  
xGm3CmbrtH8cKIVBqac0cIvBQdvqsPLAIwHNSlv2+  
zpljPa4CQx0DbkwOUKwS86uK7KmW6AMetwksYF/  
JFbyJWZQw8gg4jKgcJr1f1aSra0aqtC65b+JEZFmxLvLjvm+  
ASZt5f6VSJfk1C0jnSHux4wPELoSa8e7PlZ1B732vBTBIk74PjMR7lDfilI5Rk4RCHfi00QeMoRhM37ALACi  
/dEl05a38R3LePZ0BMKHUUYyIhA0w/  
LEnVq0N1hH7Na4HoPSgwGox0m2f0vNms1Z7A92TdBxCTbpKGAN7D+7  
atrznXb28rDPGJn7I6/W80Zv5WtSLG0vwmNR0xiq/bG7Ad07U9Yi6ihmZ9/  
AQxtkTU5n0y8o62a30fYR3wUIyzbUW9TxCoxL30G0A0fbkwvRwp0DfnVF2ev4gxjIu0bXR4LasE9JovyD  
+gM+XMxfhUtkfLgibmvztqQ50Qm+MWK8g3Id2opS8+  
nBgLu9M2B1f13XCuiIiRkaMRPQmmXlPk54iTFtZt0xG1kMvifsbGPdaa9fIX85I5eyoL2Fof2Tl0ASIYdluc  
/23mS0ioYd4t0R/1XM7IfwAn03xTitbp+rqwzVsnDDsj0so0U1Y2+  
LZN16chU0nk0iP0gGI4X205AuB9r35lfbEHEcJccry3F4CouJW+DZAJjmn0e5rzhL+  
UHG8NwgUx4zFjvcj1l/  
IOcIYwDCXP1Ha2HyRhrtYTH1Pc57Df6TAjOHWQPfZfqN5Bg4xFkaKzH4NINspMtJFXtTRnF90xUhrF04R6d  
+eq4jz2/rCXTrYHP+tG5KzMi04RLp8P0XCsdwjlDxpMTzwU/32  
Y00oEqdbc7ng6zZvXH9fzFhqsP3X+bZwVp37zUCFW/uNVImPE376Va67qSVGnqSqA3BqD/  
ZpM8C0EVY6pvvv4gZ0eM5HzblhAKGrUwnmpGCZroZHEHdG1QkUT2DQfC32fZnBKdYX575Rap9MB7VD8XIAgU  
/x2eIgc578o7rwwCBefVEjvAX7U8ufYwQBxkXnX1uvDxjhgjm/  
Raz3XSsdI1uV8gXIEYDwpG6eJR5Xw1Te9IzOuBgeGsaqTADKCKgieb8x18pBgrI4lwCLcYvZYW9Z  
/OyX3I0zcxEqscQMjhyJNYC8QMULuTH8Ib6UIV8ExSE/jI+OiGkliR86RjqUX9R/  
OWBMajhi2b81YPXN4izizBSdYHy6YnmZa03QtoJ0ayEpfj41CRdyM4R4R1TdSP1eyDCY9aZ  
+fJJ3Uwigh7iXlNsG0beXiyQ3gjei1ywUpM4ZVkguUnl9LN8BUY/kyL+g/  
lzQL4eFjBskp0eMD1UGK1AJSDx4gW8zB9CUS30jOE9j901vcxnMsEFj100sZa1H+  
kZhVeeFSwsCXIW4EIJdQs2ILz3aDP23hzvaJjd4txkCt/CAYLMTOfSA0cJ50Y9a0W9R/  
H20bwHbxDuc30kY/  
BhFITiZk2uHDMsv66r4r0oWKMczY9f5bUkgtJTxFWrcYXFPW2FvUfptUFGA8bdrJluAZ8WMXSTvWKxKoDcKG  
+  
tDFbvYVIf1jqS2AQgWse3GJeK1Gey6qE91pv6TeLmPmPCwdTU7YzUo5PyDo1hPIFi2rPsHnaS  
/Y2yqkgjXgD+RWNoa8sqOE1f7CAynGBjrOZcfY94GGI/repI6+  
uB8zM8DFKQDfLMD04A05Qm+nCQz8SSIZpJ6vp60E3ckth+CnM/

	<p>Uo6MW9R8hPEZ0yjkvJGV2KE4k+97  VSMX7eK80wNCo32JcButdMEDA015C2eyBrpppp6oHfEFR72vry0/OiHTc/G6Q/1Z+  lxKDVkZky8jri0mqk6+tJUpQbqiejnoISK+  bNVJ9KSSxLRnzx3edddGLUF6Gob626CNyM9ohb2e8afGQYwTE5ao6CU5r0kTZzvKXASWLM8gUR0A3TpFfGjBSI  /LiEzOLso5gfKHKZ+noNxU881WJi47eun1Nek36HI47QTKxb9fSZ0+JaQoaErBcw8+3  CJGONWdtu/JiDS77KKcM7hYWZ6YECPCsGUfd7Vttk4rRCvSwfsVxs5+  aW9cLPQvNTFipLVBivEmlpFYrQGGYwz0jkaL+vdjxFv28GjGbNFuF9/j0oKzEpQbUhYt01+  bkwa9iNC1gdoHuNJriSe1IVI+/mQvwm81VX0BwzUG9fc6fv9mjMi+  DyflnHB0iRkC4MsJjB18gIFXzz5uxLgS2W6mbR28f9CMdT/FjWnnPchk03ewyevC/2  PEXDrLuUA9Kp9pr4TI1J9rTsqs7zw70lSwlGM3fr1GXzMRg+v9mklg/  ea3qNYwNkp3yXoNm7RcXz5y6CaH0SN7bnB+RQE5wgYHrWofxtGjKNR03xsntJ+pMnfny53g  +WalVb7pY8bibGnWdUFN7IecSMQbhAjxdSRxNzHu3Mt/UpnhK/hr3X8/  j0Ykf2a3JRzyo4yGnml/HhcmtEmW4abD8sfF82Tz5tTnaYX2K90zD8uYPg4SSiCINGwSe+  C8QUiInPvJfr/nhGptgM35ZzBI2N/bOUFsV5ok960H00ifGBt0tc/  VB71rjGegD70eddkzXMJiZdQYq0vvIR2s9D19/85Iya5A8kBXqYjWZh7uZn3f7/  aaSGeVPV61muE4ElSe58IZRgvFXs3Ey7G6fe/A0Gr8PBNi/r/2F/s3Y122kYUBuD/3  tGGQ0z7DsYYg/F9/7  drI2kklsF1GwRyM985bRwn6aljfmyOy71K5HNAuAsmjILsP7v1zN0AlajVlyNXy+  bchE5qQjaqxsgSgg1JjNmwbTkrTK+8  t57ZjSeUEmmsXdwHg5cqW2tb4wTNRyNm72BYOSZou4adt5sQaCipsA9G5vqkTFg973AX5ndhC1Rt  +jpb5eQ3PdwJ42Uh2riPM15W7FkakfEP59guuN+0klRg00rmYi4xpSKc04rS+yCFqn3YR/  USUh9Vwn0QMKpIylm5htXORKUHMtTcdkP+npHK/x75qpXCJZ1PjXCGMAiKX07T+  ns7fy8VJBkfm06EwXvJBlaqhB0/r4ZCOMd4VOHVnpD5jrUjCefi9CCHU5dbVX09aZw+  Zw1CwaJTfatEuh3GvRA6DTmx93DlOPQvEs4Rd0VkJZs/Cfcugkt/9  ZZxgHbCFhyvHZJh9yNFiimZilcRsw7ijpiMnlDRxramndB8g5EifqBnbpH9LP5C8s42hpkz1BZcI073pRngA  /bq1Sf+Wqk5RkF/n5d0TmlcIa12y4kF4FIj1ZMGiCE+  dumh7Hf1GDTA9K0oSpvkLd0RI6tUG/Xu8hfxS2YCyJG+  CN0crMgZdGYLuEYrDbXut7an8NuG+jnqkVit4b8lH8u4ar1PKeSlDxlBSam5z/  l3UTdqu0ckxo6+6yRN61yM64a0+IBSF4Nobrm/jrwj35n2m8Y3AiELZFo/  e5Um81ZLP0FhSzsjm/F/YizqCkZlKIpyYQzdKNtLnoPMmbksXhSHwykb94XTMCXfXD7J+  IYyjHt47phmko+ueuQBQnUnKH9h99H/lywhGZpWvjZh9JEE/  ggf5laT0DKMoxDbqT6D8LYEIBTiIVI5gAsDYS2JWNbcCzS9dDIL88LbN+  b9C0B2bexVJNGGm6wDks33GWBKF7a3rqNtlUdSgY55AWgbdMDnSOVKhoxrH/  k03M6RVJdszP8l0sn5S8N8K/Cq/HMY6VBTvNcQAUAaBiEDHwJ6VexAVHgEu9p37+  qrk7qutOP/VEbuvdg+  uvkiwqIFgl05sjj1orr5n0EDBiIF23Ub9AVS9DTBDK2BUYLGWKBuk7ffa31nGMrQf6euSjD0odpEXSbV5C</p>	
76	<p>Ghm27UVzJmNCoi5N4ma29XupUq4xHi9fRiTMQjllL+x0f8tI0moHQiasYx2oGfujLUu+  E6MByAG1m+2iV0BKm/r+8ecJ82ufsQ71nApr38wxBXGuiK5hi0091vykzJDgGC0y8rla6zr  +mOIj0EMRJ92Xa4g/ufg3jEnoL/1RUQpJUqJBDsGmb0spGdKel0J1rUdFX+P+  ypKYmNzGd98c8M/  gPVtQZGHd8ogBvrLOD6s350K3ycA051jXm37clYaSuY66dfNQioTU9LnklDvsDn/  PdwMJHGrnZ/IerscZyxcSQWuIwHIga87cz040/KmW29e8ccBN4sTmJ+  aOXHxVgSLYbBVH5xjjbnv68zy4rxfXgAXfZXa0m0drpKAFFDEms8GDGAUdf040/  G744AunvMUZtfxNxcjIyoGorcKj1KqL6JSLC2Mf99DG+qJNU96MQ/  W0vnnfKORN3XZH79eZRn0IQMxDt7XbbXdT3UQEL7QRaN/KcqyD0JeW/XD+  m05LogQz15erS6Nic3wMDq0rW6Go4wIlomNVwm2c5jyoS86tkrCTLIBSJGahtW3YG/5  sqre1Ex/y+JvnLRsLP9qazntclMQbfqFQofglXXPSmNbutdh8EHOp5M45hr+  MBQLWzWwaivepVTOLLeJ5Jo+V0ggdEHa0pba/+G8LXHRVzCI57+XDu7yPE+  q30E1fFHRlDsbwZEK2wj+f3w+iPTxqi18fvw+F+XE9+  qiv1sW60nn7uEwaMpkijiaIREZCYt+wpmv/Eac0HK0ys65soSbTWADMRC1buZH+  d5logibkp6Wxzfj8M98M5ifrlh5+1  LOedQGKVMvkdoy5K1LSDwjED7mhqi059n34jn448gBmFYER+fncij+  lGYgtDbdeekpha21l6wRjIGqXoMw6SWXpgxAhdc1k/bZ+8  mMIXFI8YwGDbdcsWiP52yivdVYGDud7+  N1Qec990R306uV8e10w8vWDEqA0rF4FJQ9uEzrkuF6Cky7jGGKW/PJvgIZgAr7ecic36t2I  +G/bcoqfDBNewjKNb+PkH0+L6myS6dkgvHAOHqToNjIq/MSeN7zlrzuS84Bph0kj+  tBPHUYgBvDSngY36P6U8mDb7ejAvEmNTyUboXCcJes+U5YkviQ+4  sIpFB069h3JqNt8AnCf5NX3VtGE01OSd4gOPxARwBzbKfWXXfGxwGAMYqmSzleNvEaJ082B5CCKGHCsY</p>	

/pD8AABqPheBb6f1BvTbe9PgC6vGWg5M3FNX2nTcni0QV/iAFUD9uWncIbU65a240nB/  
PiEdzZ1WUVVTnoiH8a2XoS5JEL7mF44HerqyyCKBp2Jm3xGI/  
QbX5SSW19QWkiS9B8KjEQGYr0f2cf2USqZ16xoAIjwK46An7x5Sbs006q4RtORsI5YHIrr5E7zrgn1f1aFRss/  
/HwUyd54589UkeJvYNhNK1IYmuT/igUu97udCS2qBpTtnGeXyKAANCvrP/x4/  
pjUk7M7Lr8N7qcvGc3n+h0rzwYgGDWS8cJ370T96dyxxJTh1uVY1V6+4XwRASA+4  
f5n5z1dMZ+6HORKWem2z/D0rvLTERZU/QpGJp5Kbe+sevuz8TYJTmW0Yy0uuQ+  
LhHxL4RHIQA0ibaLPzLr8Re82EYTKi7l+mTiervvtmat8bIdpZ/SGPuTyftSp/6  
rZVqvJSLTGqzn2icvodC8hT5wzttlacR88iHhkWqDdlf9UQ/  
s8VfqdNudouPCBIzGYUVJwvHrr73TqBPcuiQ00Uyl0e/  
DZcItL47TtCWjnovQaXy1he6N06GzDbpM+WTdXK1WzY0HgBkP5fv7e/8  
PymbqKs7Tf9T0UjUhb0JFzTjh3wXQ1efcrklLj9ZebfJvI3lV7rrxjg3kfFM30pfaJEwzw6DP0HUcp5VT8  
+vvm8VGHW4s+FvJ/D3syYf+  
Iai4KR4xmqIxtXlYAI68JfsUJl32A7ZW0smIcv2qR3q9I3sslxYT01FcX3+  
etCyY8GHuT0dT/3z6xJOP5dDfxGA/AqHaV3LB4AZ90+bimKoumC7DdQysl0n0eeiBz6/  
rLPVDGS9eRa/6WwHg81xus3tT/LuzJ3YLuqu05eAzG0hRNZT9q/g6cHWxVkjn90H+  
N7E3zslomtxwjKhkL9JcuIS84N3TERH4A4xnI9Tbzhfx/wh6vNC7mkecShoXRrkushh/  
voVyQrVB161YxKnLGRX0q4CNegktRZmvEwM1X2LBjv8WT2aGkSelPvAM0uyH4Uz+  
J2bDjceMB2K0HdEaq358Xmb1NvuKZLKkU9aib+  
duxkp0KdXaAbD9UcuF0wOuvMFed981MQRnFxD+  
crUxRMRu5t564cP6qo133hMeCzGLkv0YsPIeW1fUqoHupi818FU3YZyt937DJ7uTDa4vyHtLhuuXmT3G  
+Zxps+My93vvgWSMhp6MeHAc+/Ij+ePVgAmPR+UJJEZEc5575IKB6CLNvpLEMCbq1W8+  
urprwTrBGFduXEuTpfgz0oK00l43lojF7VEey/Hqmt1t/  
xZfVrVbLmr4Wl0cM01hEa0voeu00ueLzEnnQpWPwI5F9qzcCVCmCx0za9zvNTRzXJeq0uqHuHcQbdn1BVK46/  
/QkdIVRj2uzgeXQpMd0q/  
wqh50hilH1qJikGGInNm5Lc3h5tLxXCMN1IZ30v1YaHBGESqSWBcIrgvUpCrVEORPil036f0eU8U60UM3tPM  
/Jbj40SxR3pZxwPN9Nsv/  
Jp2JsJeb3YUaYX1YJJq8iscrgZ08V65YTP8DZtbjyreq6Ey4ykS0JKTivEKlzbrrbMmkKUCulLN9RpDruh86y/  
/EASvpau73h7zWxJb1KDlk3dCihioDRcHu7tWPoQoSJP+  
NkiiTl5bUvXqSWHAXJBvvVySyVwpe2QSMTNi/  
d5q3woqqpKqZTjVILWftXrI8alyHiMMXJMb9Hmit8HpAhdiaK5+  
LwDi60lU0aMY3b2ZbZr1yaDVV1Szgh8UViuUYUZYxCKKsEG223EWdZpZX1cjhuBXyl0hI8v  
/  
QSN8fLYeyHEuEQZjxGWuuLjF3h8uZZD1dPJu2avtpQWYSk5pf91cVRmJYkNjPTvUZsyDujXcWck3M660d93G  
/Ss/g7p/hXvoNHdz5vrFxdZwsuYgWxa3uh/58BFYwKNRpJYlP57bsXoPX+Nxv+  
kpnChjb0d1Nuqb7Jk3EKlmtBd1sehWrRubofT7qIeBr9iH+deyTeoONkV3w+  
CsL4YT4erZm9Qo/zrLt0gbli0zcZqM0J/JrEBcnry/mEn6z8BEeaOXFF5ZU/CJpRYhNsYK  
//2gN5vTsr4xk/Eif06FoNNb9RefQzfp6/j7tti1qhfaJQWi1Z3/  
Dp9H36s2qPeZjDxcIK5ZAGn+CubvLz0q3w9VuczdZPC9Pq31XQNio5N+o/  
A6DXkQngEn7wYvrWkPlh7uKHWcspdBpR05F1m/  
EfMLseevmVmVItW43rFCVrzTRWgfJlV6TNutzG219uw1JTEG6wfwYU7ryvJBdM+GBrr9Zo2  
/qNOS2Y/qFkH/Y1z7t9YM3+KypntfDD34qqaWnfNICSmutfKVxiyyvUmn/TkfWiH9J+  
BCd6xW/cdpRw/bM07ACNHn3rmTvgKwWxTF6n3f07Q/1cIQK33mi2/JD/OddNs/  
tRrcV8hHJKGl+8uTvQdiU3t9/anSPqcNY/H5nqSJD9DqLZ0Zbn/p0eLknopn9H/  
CJ1267J2gMhn8iilzz/M/  
inoUwgIOuLwa8HWHoX70YjZvBWql1zVfz3e2nRE2RH90QjuZh7mKVf5h69nQW+84  
CuETWBan0c3kW5kv7M/S7bnZC4KLeMq/o0VLjRrXw4PR+j8xd6dKLdxw2AA/gHuqfu+79  
uSbbz/23W6u9yTlUaV2H30ynqZ06jSKIXBAEro6IEpMtGAafRemyl/utKYzXE/utrS2F+  
yEI++UvBjqBJuW2BcB/fmHrjxG1YVeibefF5uW9pf+pAQLhJV7mdw8C/c1cEM91G+c/hS6e  
+vzWneBeJVLtS2cbhv5W1TbszvnUdAFgfM31GmC8PtNJgPES/7Ixcmxl+4/  
izn4pGUdozER9sKLPR7B9wL8c94kqx2KRwdRNbt0kNw1bHogxVc98invn+  
E9xhALbH04nIT2h7XNZF8a+K0oSQRNUWCGWL9MGbHPgr+  
WuOihlUqMIVfIKMBWnHzsdMGHcksjtfklVKPLNLy1ZT3pcFtXBjZCa7fzz4aEQ6BSIO0vNOL0x  
/pX4WAMjh3UyVI3KLftlu4GjxDHsPrFzn8L6uQjD+  
OLD0ftUXW1NNL2iVy5Bh93JzT6ufx23K29gZAh1PUeTkEqumHbB0s12d0ln3bXA38P6uQj1ZBJKH89idELJ8/  
+FQJ/lUsGrpr3H/EW8mtTcOpK+kUjgIeUN4ioZN98ybumBYEKgRXId3eZUfrS02H3Fz+/2  
HR3lvXhIXA6c++QIREmqNtjubWruK3i0biOgEfpB5eKZd9StPwmcdoSbwIzQk4hqvrJGEP  
/M0s6MSZptn3fMAT6zS9N6KvNenW7hf/H6qIkaJQS7+9SHMWBpE2crDk+  
G2lLrAMyH8aH8SfBxv7p/HBpe9+  
zC8IjRFhIYrnHPgv0lNd805eGmzut9ynbZf2f6YvS0bZ8xrx8Bj5PxmeCCx1+

nQ6IUmsUTugJxvYw7YfLjmM3AOE+hjeTxLmBJNDDC7hOPfp1KUVha/Fd+sJ/Y4SP7USJ+  
MXaJEZXIldG4lrqFTaQmDMpnYEQa2+06FvK1k9HeJfY61Gkc9ocWMmiDsJI1G4RX3q3zg+  
TmhSF7ckcBLIL+y/pxbXKpSVdh2ojWaJPxRNPxr6WDnls6h61RMQM7AdK/2  
Ha5fzni1I6EbW4f/Dtot/  
Wcb7xQMDIvKKDALhzJSWtZD0im4c3agw9EMwozquFNxSX5ZZETgxm4KpE79zT3b2TjXrcITNe  
+KLbgdftxvOPkJug3W7mDsOIynF+  
q0lhJoRpRdcIB0kovXC8A4Tm5s2z9ZVGvJmBYcYL3duJkU1To75LwKWtQ/  
oAzvLqSrSwduwdRvV6fzhZhhJT32oEj2Xy9W2hnff+R/2LXUzTJaBH0IGe1Ft+  
nORbD9Je0z0w4yTK755GsKq2sgLDiM8ScQrbJ8I8kMgFHT1mtnwM110SUaHv+6  
GTfk19uwk81u9CwFG0sNsbJdnz6R5UfBxMXIDHgc7xL3d27v6tHb2j2ntQEev+  
02NRcBeEi6gSGiVuTWBds0mELAiXa0wh5w1A+oEQ2djrqn+  
QqKeWEQS0IQ7UGQ20s1MTUDQ8DPRu/GwAgd7+  
qRwDdhH1LIn1YJYRbv02BgetIpPQLCIdQSo6EktFaPhAMYf1R3pTkFe0Xcrv7sA+  
iQqArXYNtOvhJhu+  
y29wMQYy0Ez9q2v1iBaEZ6r15hSdKYksX0YxyEJ9QxR1fRFVWc7Wyfwx/  
nHFLirIFncBHSbU7ANGDQKe6kr85WSUGE8C46mYIV1W/Va4/  
KmyeEoXdvf7o1JZjggF5vZaU0Ks5rD808daRHGcHSuvXB5LnrzyA7gU66YtTrdIRe20tS7mM6Ft3Uf  
/dkjKFmqk5wE1S7Ubx/n9NUq0L4QPE/dVaieYfh+4f/Er/0chbhaJ15zhkeMRTx3FuRd+  
CUCRoS+QVVG1EKNK2Jzom3lFUXFxMK01lZgKGKXsy00A+otGudzpt04f6n/  
x5aoEuM0cioV5WdK0SsuWF9H7SMIplvJydzwk5jMmdS5HzzmbWXa4H1+3  
hT80ET0Tp8uIiWlUKjzSq+  
xI52eIE6xP6ncnxvM5WjZsviZYjef7bWMQY6IyVRJZg5LlnPXW/  
hPovgWUcuvz1nwiejSc8U1UXFrkoYiukvMCNnSJrNlAt55GhKKLI4kX8159KVAfBXpNIm+  
g6qwAkWCPAj6spWIw/  
kHvW3b3vZdzqxaOutfenunjg3SJx6awqV5GBaravIswdiRyg2V9Av2tUlB1SiBg2lZSосу5qm895VGpy51xG  
/  
K0azn5IVdg8LX3KCzZ7NE81FzK0iuBV9efGiW8KR4YTtbJd06xedR0vpu43Nd0eKJuBKw0Hz0  
+9oaETonUSUGChx0owK+r/  
dfvM6Nak4N910k05LpHIAyWfClnOXiD8E5zcCSUbkp3wwWv8yb6NDUE1BaZjVJ4G6H+  
iBRPS/hGJaLxwjM+8awzy7JVc27/2vZr3xsCaiDLwnx7lhCE5bIpU74m5S6M6L+Hu9G5+  
UrbnmgvUrtPIIh+DCGJZe2/kw0Am7pM6dqdRVUioDoPp+FgW03zq/z5ZB6EhCycxDybw9w  
/9H4/rxfmXKKGmcJaUuyIz0JdWDRJxpYUnfSiS0R+PW+r  
Dp2kmSP4wcZuA281UxCawxjtlUz0w2D0wNqXEoWrDYMSKNzsyXSdUBTejX501/s6T3W5/  
ar7gLUfmSpdRB92R2k03WwAVBYzeMv6M9YbN+  
UfPdV60xGEXMwHwTfBDovs4CV2c0F8fCzH1JBFG1HUcAeJMgjQYCSrc4gu80ztetN+  
bzRtOjVDXDbKyxHyzfr9f3ZRDeuSK6EiUpv1mq1wlcRtOptnsknRSp2b279Wk6nk99fJATa5gCPRvhAEI065  
+i7bE1Gw0ZsoNeUbjXcQwRYaiWSRd73tEujtq9l4G7VqttT6ednsPwnm/4ixX0mppt6hlTR  
/q5e7LSjQltT4Sr2kcc1JU063nI10XKF2+x+ti/  
e8QA2CYEEbKkANiDHT8F1C8XC1Z5erkYs5rJzF0wC158zpzZAgdUSIv+  
Abmu2sg0eH7tI6cRpD2axsC4ARQ3/hpDoINl3udIPnZ9Ryxq64/  
IozDao0C4yyRtV3SrV9ETDDTvWPPYGTI8yUyRpG+GrMDJU/buhdxDy6hhNz06uwAhMwp+  
rjogGDG+1HdfbZZFcXwC+adpYgopfIds1ZIEZ015m8BZsq/o000y+  
YK1E9jqjiIj8W9PQ10a8hse98e5f9NnaTr2yXd+  
nJzpXSgV1vRBjW2CvFS1NqXrqzLFi4MXKwktk0GX0TpWc1G+2  
A5uJ46T7zjqfTD5xFOuC42r9elQJVUuLQ0cBkFblrTquRQOEgP4qcVr5387NEFAGqlz0iEsZ807k0G9Anb0S  
+hZ/  
K4IeYRtobqTMFb6venCyMUGCRUQtPosWes8EEyaup8SP47zRnN4mV52ffdz0U6oT3KnZir+  
q3QdZaT0QB4K00slUeLKHeQvoxfl/  
RrIvEPCQDF9TJJrfGrq0gb90GV621h3a7o1lebK1W0KUoz6SMUuUdRcVkcFaYx+  
h7DjDCq5u9HywfzxC6iRCLZgR+F+  
eht4MexuJi6eB5hNBADFe2nS5mKsAnXnPtYSKwHKj7eKDnW42sB0QXBac+f+IBR/0  
yGODlhW9kl3dK+eEWfmZrLtSvlnUFhyDK5+aSSGWNTqqUljiOsoMRoW+qGYZZ2R1bJ40f/  
M3G+  
zJbzYD27bjaLQTV67fjpE8U4DWMXJsS6J3sAhsZXUSJREm4YpsWwNI8DPW7qTjiJyq4Sax0JBLZoxsK  
/EehLifQIKGXL85PYCVMVr3jM+  
BB7YfF2D0Gg0s0nPqgV7qYyVcLsTeU230pqTtyrvpSEGpw0HiLzy+  
bvLx9AhTR4Fy7MGDclKSEI2kpnIFxMlcQ60KtT+  
gDoDptKNowUNeJf3vpONQbWz1AJ9DhVFGmgKCuLo8It6isIH2LMilXzhJ2kkT4zzYX0ZtnG4Z6LiJL8X7U5n  
/F2luQgm+rOoyG1WTajCy4dpDs7EK0X712GyUM0AcmSH1EHEFIzEWlv/



77	<p>timHdbvM3f0AKDKZrtL5scrVdmGmEKm1kQPMsotZxXuZnazWTE6oarSlePhkNtJj3L0/  LRhPYKwkplYewETpVhzwXg46FFV5sMrHw2wcl8pdsqcgME4SC2eFPT5hvkz+  l12kaKwG2z0s68vNHaXKgd50hzyUC9oigwYirJNHTLiDPKfUZPI1Xzu2Q8Xcz0aGP+  z1lUgmM32bie5Y+RihmU0iBleKi5gKV8R39/  crScJtCMrfB9L7HsJEtMKLwNgaDh5xqQ02F5z19erxCJYs0N0J26q0I10W94pIWuDVBqHEVDfvgBFxX  /LHWrUGCEV7UeW8gUFjnZtfQJt48/7  KeCTLQagpmFBCacCGSRUwo6Jaor4Bpy9LLX76jv8JC8k5g8vddAce8uzwSut3oPF2XVjRGVeJvDOK5t3C4kU  +0k1UupedPGVX976Cpe7Aw7ugr3SKdiYf1HEgeHskenEutb80tdroPAn2YVP+X0+4md/  YI7kwyC3BW1ZS8ABek7Lma9fvMp8fcqkzcNnV/zJLly5F+R0/EX9g8CvRzskMgREbt+  Pus4y/Lm+EY3dDeyvhxoK/fusnKWqvjEUatUJZrRpg+9  bvTYy4CcCmXuBzpyyuaLkjcoIXfV3J0s5asX4/  IsC9XHIX15Itq2d8JlayQYx1M7oV7mJOC4Guk/  rHfjvpo9hEAcVnzKr3MNCzUQmMTTHQH0ew08AdpMv0t48+xpzKDj8+SA/GeufTaEl56pp+/  ZSKkiGW9fsRAaByy4kXRlF9Fsd1B1wM9AnuYrTyB+mEQ5JuTkeEturQcs/w/  vCZQFedZEVPl1EP2qMTsQkIH208qmIxxBkjKKfdk2Sj01k8xj4ozM7R3K6I1DbTn9cW2/  qmKPtBPdm5N6EVU28tXThGuo309VEoJM15cHqMntS/rvS1D0KON5DiWC0zfm7AsKs/  NBYu7iM0/PRz50sCvRWHSsEeFHIAuFNavneFmGqPTfhnu7W059e9j9JIqMw8l0zg+  j0iQ7pSyBuGeeekptiNK74kHEushp94qzSczm7fT3bCbF0cq2cHIMNFwBjy1dV+  Bnnnid1wuHqQreSvkN0z3Vdy5C9gq00u/UL8CHRFZMYrcq6hiqpqS4lYHhI8R0qpYk/  Oq99wu5jWpVp/NQ12vcxclx/H+Dbj4lYcAs6ycb/rko/zxs8/ohJtfrm4lDOPHETDybJhb/  x1v35nJGCX7WjphpJy+PjxVbroC5T8x/B0ondgoLbcyXngAxl267KU13irRC/rTh+  h1GFX2KzXwM1n3GrjcEntWR05HlG0IZ30z3p5QskvCzyssjo8XPdLtTzu5a5y67UR2pXvx1DG6uQ1jqMr1tI  +32iHjPqbqqBrzCKbSZQHsBQftUeXQomsD3freeFI+3  M4qZLSD8RHGVum9beEYfTOHgYCulLfS08fH60l30vSA18e8IN4GMMwqlcDT+8  ftx2qXZncm6X3XDK1E2eYx1jc3b1fGQIARPGx+xBRI8U5L35e4q3my/  AUScUaECMfP3k4PjAd2Kh/  nK8YjWVON5eNA38jj0jjm0HDFLanxvYFKWQ7bUsL65pL99LoBKnSoqN61NM9wkzESzTTwC52lpcWkJyCks8k  +2o2d43SjknLjZu94soFufW9bUYb+7+xL0pbAhPXuXtag4sP+BFz6KLgi4p2zZ/  i7G05RtGD3bKu4IP72jAcYQe7x24zZr+zcs4P0bbWHPCzr03NfspZnGcJWfbc/  LdeENaF1Ssd0hKXk78XWa/  GSWwfhlvsgqfb82ebQybk34wHCKr1JSzBiXLPr6CnGVMVNIWODCev/  huqds5L2HmQ6rDLfEGH0dSj0oNEqLV0t9490GgSg8VwZDngjeUcPz2C0iu1zHl+  k18Aj10lspKGclnBL0hD2tor1f+  R00ygbORJbV1bg7JRCwjo0b5Eeo5eL1aQNevoYvbGWPCUrwMnZa9cpHsn0/  tbMqCKdVxSfydSKp2V36tYPQbhIIPxWck+i3Z  Cqn90S+  cqBvGxA0Dx1jD5vSVkHmouyagLxiIfIDXWku3QnzvVSLqfvipJw0bRVb9ZPQXgVrXVxKXuzrxzResUINaTv2  +cNfpfFwUa59ZMUuqI5s7dDw3Ubt9NaMlvk90PwGDDy9qHe4/  PqiWN0vubnqkx0zPd1rXzQeJxM7+EhkwIS6q/27mw7bSAGA/  AvjY2XGtcsCVCSSpkQlqL3f7s22GaM7dg9p8tpg74bls0tmPFoJE17hBK0tpIL68v9J1dbv6kPhnAy0sIMO  /  tE2coXWXAQN4a3J7Vv1gCDD5LZhEBc002BhS24N5ndCIaSCF4mRdJRHZHeyM5b06o0ChXhXdhNZBCNkzJyJ  /c9hJZSpZnbvLf7wCA4T5Qmzv5EFex0ZoREivR+  XdiBdiDU7rzaeoP9uHchWMONRdeIs0I43MtjqhYWjT6GVMsWTy5tD1AYy1+  jaTz3DlYlaUkRNtJbMHoxFjYY8Im63nKBDvyoMYa+99  bcOu7gYBSehIjf0cAoSy4hE8qcYhjRyx2tPow8oMV3uPVnbGXqhtWdJbF+  ONM7w5g3DkylzYj4MelLobxECyCIyUGG+3  BhhWcxq9NMRIak2j80HeeLbXlIuT3Ag2QOdTugQTVBGQ+  jft4QnpZyONvBdtHKHuCzPwuPoS+17gOE7gheNV1DCNo0h9HNcfwRlnsQ5tgT70r9dw/  TJjqQlJhHfC60gOFxWcOCiHFxYDq9iRKuNtI61RUXeH+RLI/YdZ8pr03xs6Mg/  tPboK5lAK5gRu0YurBTqRu5NMd6doQhpIbp81Z6QfLq+  bJxFtubxKDHxbe0YmyuPlX0ctqLtEzGyjlgkNNkUDuQY0cSTnJN0resxUmXiec7rKvhlDx  /54ElGRG5+cHDEiEkf1P7HRafy2XwkCz48H07VOT1L3jIh/IMI7JmJaWlV8s/fdQHGP7+  VNUE61E2NZnmTYjvHNscfnu6+zY5quk+kg/OKC+  b9ymP68Ppw3AAAA5grVdKZRrcIZ5u1IjSzla5nu6YySou0mYG621p79ayZ/TBq3a+  QRr1SrVb2pksD6oWlrrAdeXQZ79ZgI7kksZ4LuSD8+  l5NmcpM2e45a9it6602pDpz4fmBEEnBrD+  dnMN7nDuQieEVu8kXKwghdGJ80njTyp3rIptQv6W3S2Wq/nIBae1aN4aLFWkxeajJxgV7/  pRqyYyZOIOBhHNQ27xIeJpodV+pP04vJ0ugt6FxEp7+dLs/</p>	
----	---	--

```

eNVi3oWSWYHQgBl6fYiMl3mDa1+N0pX7TsTzeF4UipqtdahSLkQojcqZ+IJkhGF0YwGi4Hz
+HvueFnwdPp4cewLpxV+rPS40ECRitRmGxezfFGzFLAkptKBidmAC4j/3
jetOffYImx5X6a1bZhLd2o52IMeXlPE6yVlF5qWw4B6EbMQ0aNlPqr+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

```

```

110         , un acompte de 50% sera
        versé lors de la réservation de ce dernier. Le solde,
        soit 50%, sera
111        versé au 3e cours, sauf accord préalable avec
112        l'éducateur.</strong></span><br /><br /></li>
113        <li><span style="font-size: 14pt;text-decoration: underline;"><
        strong>Tout abonnement commencé est dû et non
114        remboursable, quel que soit le motif de la rupture
115        anticipée.<br /><br /></strong></span></li>
116        <li><span style="font-size: 14pt;"><span style="text-decoration:
        underline;"><strong>Boris Gourdoux ne
117        pourra en aucun cas être tenu responsable</strong>
        </span> des mauvais
118        comportements et des non-acquis de mon chien en cas de non
        régularité du suivi des
119        séances de mon fait (au minimum une séance tout les 10
        jours).</span><br /><br /><br />
120    </li>
121    <li><span style="font-size: 14pt;"><span style="text-decoration:
        underline;"><strong>Toute annulation de
122        cours individuels doit être formulée au moins 24h
        avant la
123        séance</strong></span>, faute de quoi, elle sera
        facturée et/ou
124        déduite du forfait.</span><br /><br /></li>
125    <li><span style="font-size: 14pt;"><span style="text-decoration:
        underline;"><strong>TOUTE PUNITION
126        PHYSIQUE BASEE SUR LA PEUR ET/OU LA DOULEUR SUR LE
        CHIEN EST TOTALEMENT
127        PROHIBEE</strong></span>. <em>Le non-respect de l'
        animal et de cette clause mettra
128        fin immédiatement au contrat conclu entre Boris
        Gourdoux et le propriétaire du
129        chien, sans remboursement possible</em>.</span><br /><br />
130    <li><span style="font-size: 14pt;"><span style="text-decoration:
        underline;"><strong>L'UTILISATION
131        DE COLLIER ETRANGLEUR, A POINTE OU NON, COLLIER
        ELECTRIQUE ET/OU TOUT OBJET POUVANT
132        INFLIGER UNE DOULEUR AU CHIEN EST RIGOREUSEMENT
        INTERDITE</strong></span>. <em>Le
133        non-respect de l'animal et de cette clause mettra fin
        immédiatement au contrat
134        conclu entre Boris Gourdoux et le propriétaire du chien
        , sans remboursement
135        possible</em>.</span></li>
136    </ol>
137    </li>
138 </ol>
139 <p style="padding-left: 40px;"><span style="font-size: 14pt;">Choix du forfait
        :</span></p>
140 <ul>
141     <li style="list-style-type: none;">
142         <ul>
143             <?php
144                 switch ($package_number) {
145                     case 1:
146                         echo '<li><span style="font-size: 14pt;">Bilan d\'é
                            valuation : 70 &euro; / 80 CHF</span></li>';

```



```

147         break;
148     case 2:
149         echo '<li><span style="font-size: 14pt;">Bilan + 1 séance d
            \'éducation : 125 &euro; / 140 CHF</span></li>';
150         break;
151     case 3:
152         echo ' <li><span style="font-size: 14pt;">Forfait bilan + 3
            séances : 230 &euro; / 250 CHF</span></li>';
153         break;
154     case 4:
155         echo ' <li><span style="font-size: 14pt;">Forfait bilan + 6
            séances : 400 &euro; / 440 CHF</span></li>';
156         break;
157     case 5:
158         echo '<li><span style="font-size: 14pt;">Forfait bilan + 9
            séances : 520 &euro; / 560 CHF</span></li>';
159         break;
160
161     default:
162         echo "";
163         break;
164 }
165 ?>
166 </ul>
167 </li>
168 </ul>
169 <p style="padding-left: 40px;"><span style="font-size: 14pt;">Pour faire valoir
    ce que de
170     droit.</span><br /><br /><span style="font-size: 14pt;">Date et
        signature précédée de
171     la mention &laquo; lu et approuvé &laquo;</span></p>
172 <p>&nbsp;</p>
173 <h3 style="padding-left: 120px;">Lu et approuvé par <?= "$userfirstname
    $userlastname" ?></h3>
174 <p style="padding-left: 120px;">
175     <figure>
176         
177         <figcaption><?= $date ?></figcaption>
178     </figure>
179 </p>
180 </body>
181
182 </html>

```

Listing 50 – ./Sources/resources/template/iCalendar//textunderscoreappointment.php

```

1 <?php
2 /**
3  * iCalendar_appointment.php.php
4  *
5  * Template of the content of an ics file (iCalendar) used in the HelperController
    with the sendMailWithICSFile function
6  *
7  * @author Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
8  */
9 header('Content-type: text/calendar; charset=utf-8');
10 header('Content-Disposition: attachment; filename=' . $filename);
11
12 $now = new DateTime('NOW');

```

```
13
14 echo "BEGIN:VCALENDAR
15 VERSION:2.0
16 PRODID:-//douceurdechien/handcal//NONSGML v1.0//FR
17 CALSCALE:GREGORIAN
18 METHOD:PUBLISH
19 BEGIN:VEVENT
20 UID:" . md5(time()) . "
21 DTSTAMP;TZID=/Europe/Berlin:" . gmdate("Ymd\THis",$now->getTimestamp() + 60*60*2) .
    "
22 DTSTART;TZID=/Europe/Berlin:".gmdate("Ymd\THis",$start_datetime->getTimestamp() +
    60*60*2) ."
23 DTEND;TZID=/Europe/Berlin:".gmdate("Ymd\THis",$end_datetime->getTimestamp() +
    60*60*2) ."
24 SUMMARY:Rendez-vous Douceur de Chien
25 LOCATION:701 Avenue de la Bigorre, 31210 Montréjeau, France
26 ORGANIZER:MAILTO:douceurdechien@douceurdechien.com
27 END:VEVENT
28 END:VCALENDAR";
```