EcuAsist 2026

Sistema de Control de Asistencia Escolar

Version:	2026.02
Arquitectura:	PHP MVC + PDO + MySQL
Documento:	Codigo Fuente Completo + Indice
Generado:	2026-02-20

INDICE DE CONTENIDOS

1. ESTRUCTURA DE CARPETAS

```
ArchImp/
backups/
config/
config.php
database.php
controllers/
AcademicController.php AssignmentController.php
AttendanceController.php AuthController.php
BackupController.php DashboardController.php
ExportController.php InstitutionController.php
JustificationController.php NotificationController.php
ProfileController.php ReportController.php
RepresentativeController.php ScheduleController.php
SearchController.php StatsController.php
TutorController.php UserController.php
helpers/
Backup.php Logger.php Mailer.php Security.php
models/
Attendance.php ClassSchedule.php Course.php
Institution.php InstitutionShift.php Justification.php
Notification.php Representative.php Role.php
SchoolYear.php Shift.php Subject.php
TeacherAssignment.php User.php
public/
.htaccess css/global.css index.php
uploads/institution/
vendor/ [Composer: TCPDF, PhpSpreadsheet]
views/
academic/ assignments/ attendance/ auth/
backup/ dashboard/ institution/ justifications/
notifications/ partials/ profile/ reports/
representatives/ schedule/ stats/ tutor/ users/
```

2. MODULO CONFIG

config/config.php

PHP

```php
<?php
// Zona horaria Ecuador
date_default_timezone_set('America/Guayaquil');

// Definir constantes ANTES de todo
define('BASE_PATH', __DIR__ . '/..');
define('BASE_URL', 'http://localhost/ecuasistencia2026');
define('EDIT_ATTENDANCE_HOURS', 48);

// Configuración de sesión ANTES de session_start()
ini_set('session.gc_maxlifetime', 86400); // 24 horas
session_set_cookie_params([
'lifetime' => 86400, // 24 horas
'path' => '/',
'domain' => '',
'secure' => false, // true si usas HTTPS
'httponly' => true, // Protección XSS
'samesite' => 'Lax' // Protección CSRF
]);

// Iniciar sesión DESPUÉS de configurar
session_start();

// Timeout de inactividad: 30 minutos
$inactive_timeout = 1800; // 30 minutos en segundos
if (isset($_SESSION['last_activity']) &&
(time() - $_SESSION['last_activity'] > $inactive_timeout)) {
session_unset();
session_destroy();
header('Location: ' . BASE_URL . '/public/?action=login&timeout=1');
exit;
}
$_SESSION['last_activity'] = time();

// SMTP Config
define('SMTP_HOST', 'smtp.gmail.com');
define('SMTP_PORT', 587);
define('SMTP_USER', 'viktorengel@gmail.com');
define('SMTP_PASS', 'Orktvi.5/*83gM');
define('SMTP_FROM', 'noreply@ecuasist.edu.ec');
define('SMTP_NAME', 'EcuAsist2026');

require_once BASE_PATH . '/config/database.php';
require_once BASE_PATH . '/helpers/Security.php';
```

config/database.php

PHP

```php
<?php
class Database {
private $host = 'localhost';
private $db = 'ecuasistencia2026_db';
private $user = 'root';
private $pass = '';
private $charset = 'utf8mb4';
private $pdo;

public function connect() {
if ($this->pdo === null) {
$dsn = "mysql:host={$this->host};dbname={$this->db};charset={$this->charset}";
$options = [
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
PDO::ATTR_EMULATE_PREPARES => false,
];

try {
$this->pdo = new PDO($dsn, $this->user, $this->pass, $options);
} catch (PDOException $e) {
die('Error de conexión: ' . $e->getMessage());
}
}
return $this->pdo;
}
}
```

3. MODULO PUBLIC

public/index.php PHP

```php
<?php

require_once '../config/config.php';

$action = $_GET['action'] ?? 'login';

switch ($action) {
case 'register':
require_once BASE_PATH . '/controllers/AuthController.php';
$auth = new AuthController();
$auth->register();
break;

case 'login':
require_once BASE_PATH . '/controllers/AuthController.php';
$auth = new AuthController();
$auth->login();
break;

case 'logout':
require_once BASE_PATH . '/controllers/AuthController.php';
$auth = new AuthController();
$auth->logout();
break;

case 'forgot':
require_once BASE_PATH . '/controllers/AuthController.php';
$auth = new AuthController();
$auth->forgotPassword();
break;

case 'reset':
require_once BASE_PATH . '/controllers/AuthController.php';
$auth = new AuthController();
$auth->resetPassword();
break;

case 'dashboard':
require_once BASE_PATH . '/controllers/DashboardController.php';
$dash = new DashboardController();
$dash->index();
break;

case 'users':
require_once BASE_PATH . '/controllers/UserController.php';
$userCtrl = new UserController();
$userCtrl->index();
break;

case 'assign_role':
require_once BASE_PATH . '/controllers/UserController.php';
$userCtrl = new UserController();
$userCtrl->assignRole();
break;

case 'attendance_register':
require_once BASE_PATH . '/controllers/AttendanceController.php';
$attCtrl = new AttendanceController();
$attCtrl->register();
break;
```

```
61
62 case 'get_students':
63 require_once BASE_PATH . '/controllers/AttendanceController.php';
64 $attCtrl = new AttendanceController();
65 $attCtrl->getStudents();
66 break;
67
68 case 'attendance_view':
69 require_once BASE_PATH . '/controllers/AttendanceController.php';
70 $attCtrl = new AttendanceController();
71 $attCtrl->view();
72 break;
73
74 case 'my_attendance':
75 require_once BASE_PATH . '/controllers/AttendanceController.php';
76 $attCtrl = new AttendanceController();
77 $attCtrl->myAttendance();
78 break;
79
80 case 'academic':
81 require_once BASE_PATH . '/controllers/AcademicController.php';
82 $acadCtrl = new AcademicController();
83 $acadCtrl->index();
84 break;
85
86 case 'create_course':
87 require_once BASE_PATH . '/controllers/AcademicController.php';
88 $acadCtrl = new AcademicController();
89 $acadCtrl->createCourse();
90 break;
91
92 case 'create_subject':
93 require_once BASE_PATH . '/controllers/AcademicController.php';
94 $acadCtrl = new AcademicController();
95 $acadCtrl->createSubject();
96 break;
97
98 case 'enroll_students':
99 require_once BASE_PATH . '/controllers/AcademicController.php';
100 $acadCtrl = new AcademicController();
101 $acadCtrl->enrollStudents();
102 break;
103
104 case 'view_course_students':
105 require_once BASE_PATH . '/controllers/AcademicController.php';
106 $acadCtrl = new AcademicController();
107 $acadCtrl->viewCourseStudents();
108 break;
109
110 case 'reports':
111 require_once BASE_PATH . '/controllers/ReportController.php';
112 $reportCtrl = new ReportController();
113 $reportCtrl->index();
114 break;
115
116 case 'generate_pdf':
117 require_once BASE_PATH . '/controllers/ReportController.php';
118 $reportCtrl = new ReportController();
119 $reportCtrl->generatePDF();
120 break;
```

```
121
122 case 'generate_excel':
123 require_once BASE_PATH . '/controllers/ReportController.php';
124 $reportCtrl = new ReportController();
125 $reportCtrl->generateExcel();
126 break;
127
128 case 'manage_representatives':
129 require_once BASE_PATH . '/controllers/RepresentativeController.php';
130 $repCtrl = new RepresentativeController();
131 $repCtrl->manageRepresentatives();
132 break;
133
134 case 'my_children':
135 require_once BASE_PATH . '/controllers/RepresentativeController.php';
136 $repCtrl = new RepresentativeController();
137 $repCtrl->myChildren();
138 break;
139
140 case 'child_attendance':
141 require_once BASE_PATH . '/controllers/RepresentativeController.php';
142 $repCtrl = new RepresentativeController();
143 $repCtrl->childAttendance();
144 break;
145
146 case 'assignments':
147 require_once BASE_PATH . '/controllers/AssignmentController.php';
148 $assignCtrl = new AssignmentController();
149 $assignCtrl->index();
150 break;
151
152 case 'create_assignment':
153 require_once BASE_PATH . '/controllers/AssignmentController.php';
154 $assignCtrl = new AssignmentController();
155 $assignCtrl->assign();
156 break;
157
158 case 'set_tutor':
159 require_once BASE_PATH . '/controllers/AssignmentController.php';
160 $assignCtrl = new AssignmentController();
161 $assignCtrl->setTutor();
162 break;
163
164 case 'remove_tutor':
165 require_once BASE_PATH . '/controllers/AssignmentController.php';
166 $controller = new AssignmentController();
167 $controller->removeTutor();
168 break;
169
170 case 'get_course_teachers':
171 require_once BASE_PATH . '/controllers/AssignmentController.php';
172 $controller = new AssignmentController();
173 $controller->getCourseTeachers();
174 break;
175
176 case 'remove_assignment':
177 require_once BASE_PATH . '/controllers/AssignmentController.php';
178 $assignCtrl = new AssignmentController();
179 $assignCtrl->remove();
180 break;
```

```
181
182 case 'view_course_assignments':
183 require_once BASE_PATH . '/controllers/AssignmentController.php';
184 $assignCtrl = new AssignmentController();
185 $assignCtrl->viewByCourse();
186 break;
187
188 case 'profile':
189 require_once BASE_PATH . '/controllers/ProfileController.php';
190 $profileCtrl = new ProfileController();
191 $profileCtrl->view();
192 break;
193
194 case 'edit_profile':
195 require_once BASE_PATH . '/controllers/ProfileController.php';
196 $profileCtrl = new ProfileController();
197 $profileCtrl->edit();
198 break;
199
200 case 'change_password':
201 require_once BASE_PATH . '/controllers/ProfileController.php';
202 $profileCtrl = new ProfileController();
203 $profileCtrl->changePassword();
204 break;
205
206 case 'search_students':
207 require_once BASE_PATH . '/controllers/SearchController.php';
208 $searchCtrl = new SearchController();
209 $searchCtrl->searchStudents();
210 break;
211
212 case 'export_student_list':
213 require_once BASE_PATH . '/controllers/ExportController.php';
214 $exportCtrl = new ExportController();
215 $exportCtrl->exportStudentList();
216 break;
217
218 case 'stats':
219 require_once BASE_PATH . '/controllers/StatsController.php';
220 $statsCtrl = new StatsController();
221 $statsCtrl->index();
222 break;
223
224 case 'attendance_calendar':
225 require_once BASE_PATH . '/controllers/AttendanceController.php';
226 $attCtrl = new AttendanceController();
227 $attCtrl->calendar();
228 break;
229
230 case 'submit_justification':
231 require_once BASE_PATH . '/controllers/JustificationController.php';
232 $justCtrl = new JustificationController();
233 $justCtrl->submit();
234 break;
235
236 case 'my_justifications':
237 require_once BASE_PATH . '/controllers/JustificationController.php';
238 $justCtrl = new JustificationController();
239 $justCtrl->myJustifications();
240 break;
```

```
241
242 case 'pending_justifications':
243 require_once BASE_PATH . '/controllers/JustificationController.php';
244 $justCtrl = new JustificationController();
245 $justCtrl->pending();
246 break;
247
248 case 'reviewed_justifications':
249 require_once BASE_PATH . '/controllers/JustificationController.php';
250 $controller = new JustificationController();
251 $controller->reviewed();
252 break;
253
254 case 'review_justification':
255 require_once BASE_PATH . '/controllers/JustificationController.php';
256 $justCtrl = new JustificationController();
257 $justCtrl->review();
258 break;
259
260 case 'backups':
261 require_once BASE_PATH . '/controllers/BackupController.php';
262 $backupCtrl = new BackupController();
263 $backupCtrl->index();
264 break;
265
266 case 'create_backup':
267 require_once BASE_PATH . '/controllers/BackupController.php';
268 $backupCtrl = new BackupController();
269 $backupCtrl->create();
270 break;
271
272 case 'download_backup':
273 require_once BASE_PATH . '/controllers/BackupController.php';
274 $backupCtrl = new BackupController();
275 $backupCtrl->download();
276 break;
277
278 case 'cleanup_backups':
279 require_once BASE_PATH . '/controllers/BackupController.php';
280 $backupCtrl = new BackupController();
281 $backupCtrl->cleanup();
282 break;
283
284 case 'remove_role':
285 require_once BASE_PATH . '/controllers/UserController.php';
286 $userCtrl = new UserController();
287 $userCtrl->removeRole();
288 break;
289
290 case 'get_course_subjects':
291 require_once BASE_PATH . '/controllers/AttendanceController.php';
292 $controller = new AttendanceController();
293 $controller->getCourseSubjects();
294 break;
295
296 case 'get_teacher_course_subjects':
297 require_once BASE_PATH . '/controllers/AttendanceController.php';
298 $controller = new AttendanceController();
299 $controller->getTeacherCourseSubjects();
300 break;
```

```
301
302 case 'schedules':
303 require_once BASE_PATH . '/controllers/ScheduleController.php';
304 $controller = new ScheduleController();
305 $controller->index();
306 break;
307
308 case 'manage_schedule':
309 require_once BASE_PATH . '/controllers/ScheduleController.php';
310 $controller = new ScheduleController();
311 $controller->manageCourse();
312 break;
313
314 case 'delete_schedule_class':
315 require_once BASE_PATH . '/controllers/ScheduleController.php';
316 $controller = new ScheduleController();
317 $controller->deleteClass();
318 break;
319
320 case 'get_schedule_info':
321 require_once BASE_PATH . '/controllers/AttendanceController.php';
322 $controller = new AttendanceController();
323 $controller->getScheduleInfo();
324 break;
325
326 case 'get_existing_attendance':
327 require_once BASE_PATH . '/controllers/AttendanceController.php';
328 $controller = new AttendanceController();
329 $controller->getExistingAttendance();
330 break;
331
332 case 'tutor_management':
333 require_once BASE_PATH . '/controllers/AssignmentController.php';
334 $controller = new AssignmentController();
335 $controller->tutorManagement();
336 break;
337
338 case 'check_course_tutor':
339 require_once BASE_PATH . '/controllers/AssignmentController.php';
340 $controller = new AssignmentController();
341 $controller->checkCourseTutor();
342 break;
343
344 case 'get_course_subjects_schedule':
345 require_once BASE_PATH . '/controllers/ScheduleController.php';
346 $controller = new ScheduleController();
347 $controller->getCourseSubjectsSchedule();
348 break;
349
350 case 'check_schedule_conflict':
351 require_once BASE_PATH . '/controllers/ScheduleController.php';
352 $controller = new ScheduleController();
353 $controller->checkScheduleConflict();
354 break;
355
356 case 'institution':
357 require_once BASE_PATH . '/controllers/InstitutionController.php';
358 $controller = new InstitutionController();
359 $controller->index();
360 break;
```

```
361
362 case 'update_institution':
363 require_once BASE_PATH . '/controllers/InstitutionController.php';
364 $controller = new InstitutionController();
365 $controller->update();
366 break;
367
368 case 'assign_institution_shift':
369 require_once BASE_PATH . '/controllers/InstitutionController.php';
370 $controller = new InstitutionController();
371 $controller->assignShift();
372 break;
373
374 case 'remove_institution_shift':
375 require_once BASE_PATH . '/controllers/InstitutionController.php';
376 $controller = new InstitutionController();
377 $controller->removeShift();
378 break;
379
380 case 'toggle_institution_shift':
381 require_once BASE_PATH . '/controllers/InstitutionController.php';
382 (new InstitutionController())->toggleShift();
383 break;
384
385 case 'create_user':
386 require_once BASE_PATH . '/controllers/UserController.php';
387 $controller = new UserController();
388 $controller->create();
389 break;
390
391 case 'edit_user':
392 require_once BASE_PATH . '/controllers/UserController.php';
393 $controller = new UserController();
394 $controller->edit();
395 break;
396
397 case 'delete_user':
398 require_once BASE_PATH . '/controllers/UserController.php';
399 $controller = new UserController();
400 $controller->delete();
401 break;
402
403 case 'create_school_year':
404 require_once BASE_PATH . '/controllers/AcademicController.php';
405 $controller = new AcademicController();
406 $controller->createSchoolYear();
407 break;
408
409 case 'edit_school_year':
410 require_once BASE_PATH . '/controllers/AcademicController.php';
411 $controller = new AcademicController();
412 $controller->editSchoolYear();
413 break;
414
415 case 'delete_school_year':
416 require_once BASE_PATH . '/controllers/AcademicController.php';
417 $controller = new AcademicController();
418 $controller->deleteSchoolYear();
419 break;
420
```

```
case 'activate_school_year':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->activateSchoolYear();
break;

case 'deactivate_school_year':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->deactivateSchoolYear();
break;

case 'edit_course':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->editCourse();
break;

case 'delete_course':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->deleteCourse();
break;

case 'edit_subject':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->editSubject();
break;

case 'delete_subject':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->deleteSubject();
break;

case 'unenroll_student':
require_once BASE_PATH . '/controllers/AcademicController.php';
$controller = new AcademicController();
$controller->unenrollStudent();
break;

case 'delete_backup':
require_once BASE_PATH . '/controllers/BackupController.php';
$controller = new BackupController();
$controller->delete();
break;

case 'remove_representative':
require_once BASE_PATH . '/controllers/RepresentativeController.php';
$controller = new RepresentativeController();
$controller->removeRelation();
break;

case 'notifications':
require_once BASE_PATH . '/controllers/NotificationController.php';
(new NotificationController())->index();
break;

case 'notifications_mark_read':
```

```
require_once BASE_PATH . '/controllers/NotificationController.php';
(new NotificationController())->markRead();
break;

case 'notifications_mark_all':
require_once BASE_PATH . '/controllers/NotificationController.php';
(new NotificationController())->markAllRead();
break;

case 'notifications_delete':
require_once BASE_PATH . '/controllers/NotificationController.php';
(new NotificationController())->delete();
break;

case 'notifications_delete_read':
require_once BASE_PATH . '/controllers/NotificationController.php';
(new NotificationController())->deleteRead();
break;

case 'notifications_unread_json':
require_once BASE_PATH . '/controllers/NotificationController.php';
(new NotificationController())->getUnread();
break;

case 'tutor_course_attendance':
require_once BASE_PATH . '/controllers/TutorController.php';
(new TutorController())->courseAttendance();
break;

case 'tutor_course_attendance_ajax':
require_once BASE_PATH . '/controllers/TutorController.php';
(new TutorController())->ajax();
break;

default:
header('Location: ?action=login');
}
```

public/.htaccess — Apache

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /ecuasistencia2026/public/

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?action=$1 [QSA,L]
</IfModule>
```

4. MODULO CONTROLLERS (18 archivos)

4.1 — AcademicController.php

controllers/AcademicController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/Subject.php';
require_once BASE_PATH . '/models/SchoolYear.php';
require_once BASE_PATH . '/models/Shift.php';
require_once BASE_PATH . '/models/User.php';

class AcademicController {
private $courseModel;
private $subjectModel;
private $schoolYearModel;
private $shiftModel;
private $userModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$db = new Database();
$this->courseModel = new Course($db);
$this->subjectModel = new Subject($db);
$this->schoolYearModel = new SchoolYear($db);
$this->shiftModel = new Shift($db);
$this->userModel = new User($db);
}

public function index() {
$courses = $this->courseModel->getAll();
$subjects = $this->subjectModel->getAll();
$schoolYears = $this->schoolYearModel->getAll();
$shifts = $this->shiftModel->getAll();

include BASE_PATH . '/views/academic/index.php';
}

public function createCourse() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$activeYear = $this->schoolYearModel->getActive();

if (!$activeYear) {
header('Location: ?action=academic&error=no_active_year');
exit;
}

$data = [
':institution_id' => $_SESSION['institution_id'],
':school_year_id' => $activeYear['id'],
':name' => html_entity_decode(Security::sanitize($_POST['name']), ENT_QUOTES, 'UTF-8'),
':grade_level' => html_entity_decode(Security::sanitize($_POST['grade_level']), ENT_QUOTES, 'UTF-8'),
':parallel' => Security::sanitize($_POST['parallel']),
':shift_id' => (int)$_POST['shift_id']
];

$this->courseModel->create($data);
header('Location: ?action=academic&course_success=1');
exit;
}
```

```
}

public function createSubject() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$data = [
':institution_id' => $_SESSION['institution_id'],
':name' => Security::sanitize($_POST['name']),
':code' => Security::sanitize($_POST['code'])
];

$this->subjectModel->create($data);
header('Location: ?action=academic&subject_success=1');
exit;
}
}

// ============================================
// CRUD CURSOS
// ============================================

public function editCourse() {
$courseId = (int)$_GET['id'];
$course = $this->courseModel->findById($courseId);

if (!$course || $course['institution_id'] != $_SESSION['institution_id']) {
header('Location: ?action=academic&error=course_not_found');
exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
$shifts = $this->shiftModel->getAll();
include BASE_PATH . '/views/academic/course_edit.php';
return;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$data = [
'id' => $courseId,
'name' => html_entity_decode(Security::sanitize($_POST['name']), ENT_QUOTES, 'UTF-8'),
'grade_level' => html_entity_decode(Security::sanitize($_POST['grade_level']), ENT_QUOTES, 'UTF-8'),
'parallel' => Security::sanitize($_POST['parallel']),
'shift_id' => (int)$_POST['shift_id']
];

if ($this->courseModel->update($data)) {
header('Location: ?action=academic&course_updated=1');
exit;
} else {
$errors[] = "Error al actualizar el curso";
$shifts = $this->shiftModel->getAll();
include BASE_PATH . '/views/academic/course_edit.php';
}
}
}

public function deleteCourse() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$courseId = (int)$_POST['course_id'];
$course = $this->courseModel->findById($courseId);

```

```
if (!$course || $course['institution_id'] != $_SESSION['institution_id']) {
header('Location: ?action=academic&error=course_not_found');
exit;
}

// Verificar si tiene estudiantes matriculados
$students = $this->courseModel->getEnrolledStudents($courseId);
if (count($students) > 0) {
header('Location: ?action=academic&error=course_has_students');
exit;
}

// Verificar si tiene asignaciones docentes
$db = new Database();
$stmt = $db->connect()->prepare("SELECT COUNT(*) as count FROM teacher_assignments WHERE course_id = :id");
$stmt->execute([':id' => $courseId]);
$result = $stmt->fetch();

if ($result['count'] > 0) {
header('Location: ?action=academic&error=course_has_assignments');
exit;
}

if ($this->courseModel->delete($courseId)) {
header('Location: ?action=academic&course_deleted=1');
exit;
} else {
header('Location: ?action=academic&error=delete_failed');
exit;
}
}
}

// ============================================
// CRUD ASIGNATURAS
// ============================================

public function editSubject() {
$subjectId = (int)$_GET['id'];
$subject = $this->subjectModel->findById($subjectId);

if (!$subject || $subject['institution_id'] != $_SESSION['institution_id']) {
header('Location: ?action=academic&error=subject_not_found');
exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
include BASE_PATH . '/views/academic/subject_edit.php';
return;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$data = [
'id' => $subjectId,
'name' => Security::sanitize($_POST['name']),
'code' => Security::sanitize($_POST['code'])
];

if ($this->subjectModel->update($data)) {
header('Location: ?action=academic&subject_updated=1');
```

```
exit;
} else {
$errors[] = "Error al actualizar la asignatura";
include BASE_PATH . '/views/academic/subject_edit.php';
}
}
}

public function deleteSubject() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$subjectId = (int)$_POST['subject_id'];
$subject = $this->subjectModel->findById($subjectId);

if (!$subject || $subject['institution_id'] != $_SESSION['institution_id']) {
header('Location: ?action=academic&error=subject_not_found');
exit;
}

// Verificar si tiene asignaciones docentes
$db = new Database();
$stmt = $db->connect()->prepare("SELECT COUNT(*) as count FROM teacher_assignments WHERE subject_id = :id");
$stmt->execute([':id' => $subjectId]);
$result = $stmt->fetch();

if ($result['count'] > 0) {
header('Location: ?action=academic&error=subject_has_assignments');
exit;
}

if ($this->subjectModel->delete($subjectId)) {
header('Location: ?action=academic&subject_deleted=1');
exit;
} else {
header('Location: ?action=academic&error=delete_failed');
exit;
}
}
}

public function enrollStudents() {
$activeYear = $this->schoolYearModel->getActive();
$courses = $this->courseModel->getAll();
$availableStudents = $this->userModel->getStudentsNotEnrolled($activeYear['id']);
$allStudents = $this->userModel->getByRole('estudiante');

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$courseId = (int)$_POST['course_id'];
$studentIds = $_POST['student_ids'] ?? [];

$enrolled = 0;
$errors = 0;

foreach ($studentIds as $studentId) {
if ($this->courseModel->enrollStudent($courseId, (int)$studentId, $activeYear['id'])) {
$enrolled++;
} else {
$errors++;
}
}

```

```
header('Location: ?action=enroll_students&enrolled=' . $enrolled . '&errors=' . $errors);
exit;
}

include BASE_PATH . '/views/academic/enroll.php';
}

public function unenrollStudent() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$studentId = (int)$_POST['student_id'];
$activeYear = $this->schoolYearModel->getActive();

if (!$activeYear) {
header('Location: ?action=enroll_students&error=no_active_year');
exit;
}

// Verificar que el estudiante existe y está matriculado
$course = $this->userModel->getStudentCourse($studentId, $activeYear['id']);

if (!$course) {
header('Location: ?action=enroll_students&error=not_enrolled');
exit;
}

// Retirar estudiante
if ($this->courseModel->unenrollStudent($studentId, $activeYear['id'])) {
header('Location: ?action=enroll_students&unenrolled=1');
exit;
} else {
header('Location: ?action=enroll_students&error=unenroll_failed');
exit;
}
}
}

public function viewCourseStudents() {
$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Location: ?action=academic');
exit;
}

$course = $this->courseModel->getAll();
$course = array_filter($course, fn($c) => $c['id'] == $courseId);
$course = reset($course);

$students = $this->courseModel->getEnrolledStudents($courseId);

include BASE_PATH . '/views/academic/course_students.php';
}

// ============================================
// CRUD AÑOS LECTIVOS
// ============================================

public function createSchoolYear() {
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
include BASE_PATH . '/views/academic/school_year_create.php';
```

```
return;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$errors = [];

// Validaciones
if (empty($_POST['name'])) {
$errors[] = "El nombre es obligatorio";
}
if (empty($_POST['start_date'])) {
$errors[] = "La fecha de inicio es obligatoria";
}
if (empty($_POST['end_date'])) {
$errors[] = "La fecha de fin es obligatoria";
}

if (!empty($_POST['start_date']) && !empty($_POST['end_date'])) {
if (strtotime($_POST['end_date']) <= strtotime($_POST['start_date'])) {
$errors[] = "La fecha de fin debe ser posterior a la fecha de inicio";
}

// Verificar solapamiento de fechas
if ($this->schoolYearModel->checkOverlap($_POST['start_date'], $_POST['end_date'])) {
$errors[] = "Ya existe un año lectivo con fechas que se solapan";
}
}

if (!empty($errors)) {
include BASE_PATH . '/views/academic/school_year_create.php';
return;
}

// Crear año lectivo
$data = [
'institution_id' => $_SESSION['institution_id'],
'name' => Security::sanitize($_POST['name']),
'start_date' => $_POST['start_date'],
'end_date' => $_POST['end_date'],
'is_active' => isset($_POST['is_active']) ? 1 : 0
];

// Si se marca como activo, desactivar los demás ANTES de insertar
if ($data['is_active']) {
$this->schoolYearModel->activate(0);
}

// create() ahora devuelve el ID insertado (o 0 si falla)
$newId = $this->schoolYearModel->create($data);

if ($newId > 0) {
// Activar el nuevo año lectivo con el ID real
if ($data['is_active']) {
$this->schoolYearModel->activate($newId);
}
header('Location: ?action=academic&sy_created=1');
exit;
} else {
$errors[] = "Error al crear el año lectivo";
include BASE_PATH . '/views/academic/school_year_create.php';
```

```
}
}
}

public function editSchoolYear() {
$yearId = (int)$_GET['id'];
$year = $this->schoolYearModel->findById($yearId);

if (!$year) {
header('Location: ?action=academic&error=year_not_found');
exit;
}

// Verificar institución
if ($year['institution_id'] != $_SESSION['institution_id']) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
include BASE_PATH . '/views/academic/school_year_edit.php';
return;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$errors = [];

// Validaciones
if (empty($_POST['name'])) {
$errors[] = "El nombre es obligatorio";
}
if (empty($_POST['start_date'])) {
$errors[] = "La fecha de inicio es obligatoria";
}
if (empty($_POST['end_date'])) {
$errors[] = "La fecha de fin es obligatoria";
}

if (!empty($_POST['start_date']) && !empty($_POST['end_date'])) {
if (strtotime($_POST['end_date']) <= strtotime($_POST['start_date'])) {
$errors[] = "La fecha de fin debe ser posterior a la fecha de inicio";
}

// Verificar solapamiento (excluyendo el actual)
if ($this->schoolYearModel->checkOverlap($_POST['start_date'], $_POST['end_date'], $yearId)) {
$errors[] = "Ya existe otro año lectivo con fechas que se solapan";
}
}

if (!empty($errors)) {
include BASE_PATH . '/views/academic/school_year_edit.php';
return;
}

// Actualizar año lectivo
$data = [
'id' => $yearId,
'name' => Security::sanitize($_POST['name']),
'start_date' => $_POST['start_date'],
'end_date' => $_POST['end_date']
];
```

```
421
422 if ($this->schoolYearModel->update($data)) {
423 header('Location: ?action=academic&sy_updated=1');
424 exit;
425 } else {
426 $errors[] = "Error al actualizar el año lectivo";
427 include BASE_PATH . '/views/academic/school_year_edit.php';
428 }
429 }
430 }
431
432 public function deleteSchoolYear() {
433 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
434 $yearId = (int)$_POST['year_id'];
435 $year = $this->schoolYearModel->findById($yearId);
436
437 if (!$year) {
438 header('Location: ?action=academic&error=year_not_found');
439 exit;
440 }
441
442 // Verificar institución
443 if ($year['institution_id'] != $_SESSION['institution_id']) {
444 die('Acceso denegado');
445 }
446
447 // No permitir eliminar año activo
448 if ($year['is_active'] == 1) {
449 header('Location: ?action=academic&error=cannot_delete_active');
450 exit;
451 }
452
453 if ($this->schoolYearModel->delete($yearId)) {
454 header('Location: ?action=academic&sy_deleted=1');
455 exit;
456 } else {
457 header('Location: ?action=academic&error=has_courses');
458 exit;
459 }
460 }
461 }
462
463 public function activateSchoolYear() {
464 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
465 $yearId = (int)$_POST['year_id'];
466 $year = $this->schoolYearModel->findById($yearId);
467
468 if (!$year) {
469 header('Location: ?action=academic&error=year_not_found');
470 exit;
471 }
472
473 // Verificar institución
474 if ($year['institution_id'] != $_SESSION['institution_id']) {
475 die('Acceso denegado');
476 }
477
478 if ($this->schoolYearModel->activate($yearId)) {
479 header('Location: ?action=academic&sy_activated=1');
480 exit;
```

```
} else {
header('Location: ?action=academic&error=activate_failed');
exit;
}
}
}

public function deactivateSchoolYear() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$yearId = (int)$_POST['year_id'];
$year = $this->schoolYearModel->findById($yearId);

if (!$year) {
header('Location: ?action=academic&error=year_not_found');
exit;
}

// Verificar institución
if ($year['institution_id'] != $_SESSION['institution_id']) {
die('Acceso denegado');
}

if ($this->schoolYearModel->deactivate($yearId)) {
header('Location: ?action=academic&sy_deactivated=1');
exit;
} else {
header('Location: ?action=academic&error=deactivate_failed');
exit;
}
}
}
}
```

4.2 — AssignmentController.php

controllers/AssignmentController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/TeacherAssignment.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/Subject.php';
require_once BASE_PATH . '/models/SchoolYear.php';
require_once BASE_PATH . '/models/User.php';

class AssignmentController {
private $assignmentModel;
private $courseModel;
private $subjectModel;
private $schoolYearModel;
private $userModel;

public function __construct() {
$db = new Database();
$this->assignmentModel = new TeacherAssignment($db);
$this->courseModel = new Course($db);
$this->subjectModel = new Subject($db);
$this->schoolYearModel = new SchoolYear($db);
$this->userModel = new User($db);
}

public function index() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$courses = $this->courseModel->getAll();
$subjects = $this->subjectModel->getAll();
$teachers = $this->userModel->getByRole('docente');
$activeYear = $this->schoolYearModel->getActive();
$assignments = $this->assignmentModel->getAll();

include BASE_PATH . '/views/assignments/index.php';
}

public function assign() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$activeYear = $this->schoolYearModel->getActive();

$data = [
':teacher_id' => (int)$_POST['teacher_id'],
':course_id' => (int)$_POST['course_id'],
':subject_id' => (int)$_POST['subject_id'],
':school_year_id' => $activeYear['id'],
':is_tutor' => 0
];

$result = $this->assignmentModel->assign($data);

if ($result['success']) {
header('Location: ?action=assignments&success=1');
```

```
} else {
header('Location: ?action=assignments&error=' . urlencode($result['message']));
}
exit;
}
}

public function setTutor() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$activeYear = $this->schoolYearModel->getActive();
$courseId = (int)$_POST['course_id'];
$teacherId = (int)$_POST['teacher_id'];

$result = $this->assignmentModel->setTutor($courseId, $teacherId, $activeYear['id']);

if ($result['success']) {
header('Location: ?action=tutor_management&tutor_success=1');
} else {
header('Location: ?action=tutor_management&tutor_error=' . urlencode($result['message']));
}
exit;
}
}

public function removeTutor() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$activeYear = $this->schoolYearModel->getActive();
$courseId = (int)$_POST['course_id'];

$db = new Database();
$sql = "UPDATE teacher_assignments
SET is_tutor = 0
WHERE course_id = :course_id
AND school_year_id = :school_year_id";

$stmt = $db->connect()->prepare($sql);
$stmt->execute([
':course_id' => $courseId,
':school_year_id' => $activeYear['id']
]);

header('Location: ?action=tutor_management&tutor_removed=1');
exit;
}
}

public function remove() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
```

```
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$assignmentId = (int)$_POST['assignment_id'];
$result = $this->assignmentModel->remove($assignmentId);

if ($result['success']) {
header('Location: ?action=assignments&removed=1');
} else {
header('Location: ?action=assignments&error=' . urlencode($result['message']));
}
exit;
}
}

public function viewByCourse() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Location: ?action=assignments');
exit;
}

$courses = $this->courseModel->getAll();
$course = array_filter($courses, fn($c) => $c['id'] == $courseId);
$course = reset($course);

$activeYear = $this->schoolYearModel->getActive();
$assignments = $this->assignmentModel->getByCourse($courseId);
$tutor = $this->assignmentModel->getTutorByCourse($courseId, $activeYear['id']);

include BASE_PATH . '/views/assignments/view_course.php';
}

public function getCourseTeachers() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['error' => 'No autenticado']);
exit;
}

$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

// Obtener año lectivo activo
$db = new Database();
$sqlYear = "SELECT id FROM school_years WHERE is_active = 1 LIMIT 1";
$stmtYear = $db->connect()->query($sqlYear);
$activeYear = $stmtYear->fetch();

```

```
if (!$activeYear) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

// Solo docentes del curso que:
// 1. NO son tutores de otro curso
// 2. NO son tutores del curso actual
$sql = "SELECT DISTINCT ta.teacher_id,
CONCAT(u.last_name, ' ', u.first_name) as teacher_name
FROM teacher_assignments ta
INNER JOIN users u ON ta.teacher_id = u.id
WHERE ta.course_id = :course_id
AND ta.teacher_id NOT IN (
SELECT teacher_id
FROM teacher_assignments
WHERE is_tutor = 1
AND school_year_id = :school_year_id
)
ORDER BY u.last_name, u.first_name";

$stmt = $db->connect()->prepare($sql);
$stmt->execute([
':course_id' => $courseId,
':school_year_id' => $activeYear['id']
]);
$teachers = $stmt->fetchAll(PDO::FETCH_ASSOC);

header('Content-Type: application/json');
echo json_encode($teachers);
exit;
}

public function tutorManagement() {
$courses = $this->courseModel->getAll();
$assignments = $this->assignmentModel->getAll();

include BASE_PATH . '/views/assignments/tutor.php';
}

public function checkCourseTutor() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(null);
exit;
}

$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Content-Type: application/json');
echo json_encode(null);
exit;
}

$db = new Database();
require_once BASE_PATH . '/models/SchoolYear.php';
$yearModel = new SchoolYear($db);
$activeYear = $yearModel->getActive();
```

```
241
242 $sql = "SELECT CONCAT(u.last_name, ' ', u.first_name) as tutor_name
243 FROM teacher_assignments ta
244 INNER JOIN users u ON ta.teacher_id = u.id
245 WHERE ta.course_id = :course_id
246 AND ta.school_year_id = :school_year_id
247 AND ta.is_tutor = 1
248 LIMIT 1";
249
250 $stmt = $db->connect()->prepare($sql);
251 $stmt->execute([
252 ':course_id' => $courseId,
253 ':school_year_id' => $activeYear['id']
254 ]);
255
256 $tutor = $stmt->fetch(PDO::FETCH_ASSOC);
257
258 header('Content-Type: application/json');
259 echo json_encode($tutor);
260 exit;
261 }
262 }
```

4.3 — AttendanceController.php

controllers/AttendanceController.php PHP

```php
<?php
// ARCHIVO COMPLETO - Reemplaza TODO el archivo
// controllers/AttendanceController.php

require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/Subject.php';
require_once BASE_PATH . '/models/SchoolYear.php';
require_once BASE_PATH . '/models/Shift.php';
require_once BASE_PATH . '/models/Notification.php';

class AttendanceController {
private $attendanceModel;
private $courseModel;
private $subjectModel;
private $schoolYearModel;
private $shiftModel;
private $notificationModel;

public function __construct() {
Security::requireLogin();
$db = new Database();

$this->attendanceModel = new Attendance($db);
$this->courseModel = new Course($db);
$this->subjectModel = new Subject($db);
$this->schoolYearModel = new SchoolYear($db);
$this->shiftModel = new Shift($db);
$this->notificationModel = new Notification($db);
}

private function isWithin48BusinessHours($date) {
$targetDate = new DateTime($date);
$today = new DateTime();
$today->setTime(0, 0, 0);

if ($targetDate->format('Y-m-d') === $today->format('Y-m-d')) {
return true;
}
if ($targetDate > $today) {
return false;
}

$businessHours = 0;
$current = clone $targetDate;

while ($current < $today) {
$current->modify('+1 day');
$dayOfWeek = (int)$current->format('N');
if ($dayOfWeek < 6) {
$businessHours += 24;
}
}

return $businessHours <= EDIT_ATTENDANCE_HOURS;
}

private function getMaxEditDate() {
$today = new DateTime();
```

```
$businessHoursNeeded = EDIT_ATTENDANCE_HOURS;
$current = clone $today;

while ($businessHoursNeeded > 0) {
$current->modify('+1 day');
$dayOfWeek = (int)$current->format('N');
if ($dayOfWeek < 6) {
$businessHoursNeeded -= 24;
}
}

return $current->format('Y-m-d');
}

public function register() {
if (!Security::hasRole(['docente', 'autoridad'])) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$activeYear = $this->schoolYearModel->getActive();
$date = $_POST['date'];
$today = date('Y-m-d');

if (strtotime($date) > strtotime($today)) {
header('Location: ?action=attendance_register&error=future');
exit;
}

if (!$this->isWithin48BusinessHours($date)) {
header('Location: ?action=attendance_register&error=toolate');
exit;
}

$scheduleId = (int)$_POST['schedule_id'];

// Obtener datos del horario incluyendo nombre de asignatura
$db = new Database();
$pdo = $db->connect();
$sql = "SELECT cs.*, s.name as subject_name
FROM class_schedule cs
INNER JOIN subjects s ON cs.subject_id = s.id
WHERE cs.id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':id' => $scheduleId]);
$scheduleData = $stmt->fetch();

$subjectName = $scheduleData['subject_name'] ?? 'clase';
$dateFormatted = date('d/m/Y', strtotime($date));

foreach ($_POST as $key => $value) {
if (strpos($key, 'status_') === 0) {
$studentId = (int)str_replace('status_', '', $key);
$status = Security::sanitize($value);
$observation = Security::sanitize($_POST['obs_' . $studentId] ?? '');

$data = [
':student_id' => $studentId,
':course_id' => $scheduleData['course_id'],
':subject_id' => $scheduleData['subject_id'],
```

```
':teacher_id' => $_SESSION['user_id'],
':school_year_id' => $activeYear['id'],
':shift_id' => 1,
':date' => $date,
':hour_period' => $scheduleData['period_number'] . 'ra hora',
':status' => $status,
':observation' => $observation
];

$this->attendanceModel->create($data);

// Notificar al estudiante si fue marcado ausente
// El enlace lleva directo a su asistencia para que pueda justificar
if ($status === 'ausente') {
$this->notificationModel->create(
$studentId,
'■ Ausencia registrada',
"Se registró una ausencia el {$dateFormatted} en {$subjectName}. Puedes justificarla.",
'ausente',
'?action=my_attendance'
);
}

// Notificar al estudiante si fue marcado con tardanza
if ($status === 'tardanza') {
$this->notificationModel->create(
$studentId,
'■ Tardanza registrada',
"Se registró una tardanza el {$dateFormatted} en {$subjectName}.",
'warning',
'?action=my_attendance'
);
}
}
}

header('Location: ?action=attendance_register&success=1');
exit;
}

// Obtener clases del docente para hoy
$activeYear = $this->schoolYearModel->getActive();

$db = new Database();
require_once BASE_PATH . '/models/ClassSchedule.php';
$scheduleModel = new ClassSchedule($db);

$todayClasses = $scheduleModel->getTeacherScheduleToday($_SESSION['user_id'], $activeYear['id']);
$minDate = $this->calculateMinDate();
$maxEditDate = $this->getMaxEditDate();

include BASE_PATH . '/views/attendance/register.php';
}

private function getTeacherCourses($teacherId) {
$sql = "SELECT DISTINCT c.*, sh.name as shift_name
FROM courses c
INNER JOIN teacher_assignments ta ON c.id = ta.course_id
INNER JOIN shifts sh ON c.shift_id = sh.id
WHERE ta.teacher_id = :teacher_id
```

```
ORDER BY c.name";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':teacher_id' => $teacherId]);
return $stmt->fetchAll();
}

private function calculateMinDate() {
$date = new DateTime();
$hoursBack = 0;

while ($hoursBack < 48) {
$date->modify('-1 day');
$dayOfWeek = $date->format('N');
if ($dayOfWeek >= 1 && $dayOfWeek <= 5) {
$hoursBack += 24;
}
}

return $date->format('Y-m-d');
}

public function getStudents() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$courseId = (int)$_POST['course_id'];
$students = $this->courseModel->getStudents($courseId);
header('Content-Type: application/json');
echo json_encode($students);
exit;
}
}

public function view() {
if (!Security::hasRole(['docente', 'inspector', 'autoridad'])) {
die('Acceso denegado');
}

$courses = $this->courseModel->getAll();
$attendances = [];

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$courseId = (int)$_POST['course_id'];
$date = $_POST['date'];
$attendances = $this->attendanceModel->getByCourse($courseId, $date);
}

include BASE_PATH . '/views/attendance/view.php';
}

public function myAttendance() {
if (!Security::hasRole('estudiante')) {
die('Acceso denegado');
}

$studentId = $_SESSION['user_id'];
$attendances = $this->attendanceModel->getByStudent($studentId);

include BASE_PATH . '/views/attendance/my_attendance.php';
}
```

```
241
242 public function calendar() {
243 if (!Security::hasRole(['docente', 'autoridad', 'inspector'])) {
244 die('Acceso denegado');
245 }
246
247 $courses = $this->courseModel->getAll();
248
249 if (isset($_GET['course_id'])) {
250 $courseId = (int)$_GET['course_id'];
251 $month = $_GET['month'] ?? date('Y-m');
252
253 list($year, $monthNum) = explode('-', $month);
254
255 $monthNames = [
256 '01' => 'Enero', '02' => 'Febrero', '03' => 'Marzo',
257 '04' => 'Abril', '05' => 'Mayo', '06' => 'Junio',
258 '07' => 'Julio', '08' => 'Agosto', '09' => 'Septiembre',
259 '10' => 'Octubre', '11' => 'Noviembre', '12' => 'Diciembre'
260 ];
261
262 $monthName = $monthNames[$monthNum];
263 $prevMonth = date('Y-m', strtotime($month . '-01 -1 month'));
264 $nextMonth = date('Y-m', strtotime($month . '-01 +1 month'));
265 $firstDay = date('w', strtotime($year . '-' . $monthNum . '-01'));
266 $daysInMonth = date('t', strtotime($year . '-' . $monthNum . '-01'));
267
268 $calendarDays = [];
269
270 for ($i = 0; $i < $firstDay; $i++) {
271 $calendarDays[] = ['day' => '', 'classes' => ''];
272 }
273
274 for ($day = 1; $day <= $daysInMonth; $day++) {
275 $date = $year . '-' . $monthNum . '-' . str_pad($day, 2, '0', STR_PAD_LEFT);
276 $dayOfWeek = date('w', strtotime($date));
277 $classes = [];
278
279 if ($date == date('Y-m-d')) $classes[] = 'today';
280 if ($dayOfWeek == 0 || $dayOfWeek == 6) $classes[] = 'weekend';
281
282 $attendance = $this->attendanceModel->getDayStats($courseId, $date);
283
284 if ($attendance && $attendance['total'] > 0) {
285 $classes[] = 'has-attendance';
286 }
287
288 $calendarDays[] = [
289 'day' => $day,
290 'classes' => implode(' ', $classes),
291 'attendance' => $attendance
292 ];
293 }
294
295 include BASE_PATH . '/views/attendance/calendar.php';
296 } else {
297 include BASE_PATH . '/views/attendance/calendar.php';
298 }
299 }
300
```

```
public function getCourseSubjects() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['error' => 'No autenticado']);
exit;
}

$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

$sql = "SELECT DISTINCT ta.subject_id, s.name as subject_name
FROM teacher_assignments ta
INNER JOIN subjects s ON ta.subject_id = s.id
WHERE ta.course_id = :course_id
ORDER BY s.name";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':course_id' => $courseId]);
$subjects = $stmt->fetchAll(PDO::FETCH_ASSOC);

header('Content-Type: application/json');
echo json_encode($subjects);
exit;
}

public function getTeacherCourseSubjects() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['error' => 'No autenticado']);
exit;
}

$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

if (Security::hasRole('autoridad')) {
$sql = "SELECT DISTINCT ta.subject_id, s.name as subject_name
FROM teacher_assignments ta
INNER JOIN subjects s ON ta.subject_id = s.id
WHERE ta.course_id = :course_id
ORDER BY s.name";
$params = [':course_id' => $courseId];
} else {
$sql = "SELECT DISTINCT ta.subject_id, s.name as subject_name
FROM teacher_assignments ta
INNER JOIN subjects s ON ta.subject_id = s.id
WHERE ta.course_id = :course_id
AND ta.teacher_id = :teacher_id
ORDER BY s.name";
```

```
$params = [':course_id' => $courseId, ':teacher_id' => $_SESSION['user_id']];
}

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute($params);
$subjects = $stmt->fetchAll(PDO::FETCH_ASSOC);

header('Content-Type: application/json');
echo json_encode($subjects);
exit;
}

public function getScheduleInfo() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['error' => 'No autenticado']);
exit;
}

$scheduleId = (int)($_GET['schedule_id'] ?? 0);

if (!$scheduleId) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

$sql = "SELECT * FROM class_schedule WHERE id = :id";
$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':id' => $scheduleId]);
$schedule = $stmt->fetch(PDO::FETCH_ASSOC);

header('Content-Type: application/json');
echo json_encode($schedule);
exit;
}

public function getExistingAttendance() {
header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
echo json_encode([]);
exit;
}

$scheduleId = (int)($_GET['schedule_id'] ?? 0);
$date = $_GET['date'] ?? date('Y-m-d');

if (!$scheduleId) {
echo json_encode([]);
exit;
}

$db = new Database();
$pdo = $db->connect();
$sql = "SELECT * FROM class_schedule WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':id' => $scheduleId]);
```

```php
$schedule = $stmt->fetch();

if (!$schedule) {
echo json_encode([]);
exit;
}

$sql2 = "SELECT student_id, status, observation
FROM attendances
WHERE course_id = :course_id
AND subject_id = :subject_id
AND date = :date
AND hour_period = :hour_period";
$stmt2 = $pdo->prepare($sql2);
$stmt2->execute([
':course_id' => $schedule['course_id'],
':subject_id' => $schedule['subject_id'],
':date' => $date,
':hour_period' => $schedule['period_number'] . 'ra hora'
]);

echo json_encode($stmt2->fetchAll(PDO::FETCH_ASSOC));
exit;
}
}
```

4.4 — AuthController.php

controllers/AuthController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/helpers/Mailer.php';

class AuthController {
private $userModel;

public function __construct() {
$db = new Database();
$this->userModel = new User($db);
}

public function register() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
if (!Security::validateToken($_POST['csrf_token'] ?? '')) {
die('Token inválido');
}

$data = [
'institution_id' => 1,
'username' => Security::sanitize($_POST['username']),
'email' => Security::sanitize($_POST['email']),
'password' => $_POST['password'],
'first_name' => Security::sanitize($_POST['first_name']),
'last_name' => Security::sanitize($_POST['last_name']),
'dni' => Security::sanitize($_POST['dni'] ?? ''),
'phone' => Security::sanitize($_POST['phone'] ?? '')
];

if ($this->userModel->create($data)) {
header('Location: ' . BASE_URL . '/public/index.php?action=login&registered=1');
exit;
}
}

include BASE_PATH . '/views/auth/register.php';
}

public function login() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
if (!Security::validateToken($_POST['csrf_token'] ?? '')) {
die('Token inválido');
}

$emailOrUsername = Security::sanitize($_POST['email']);
$password = $_POST['password'];

// Buscar por email O username
$user = $this->userModel->findByEmailOrUsername($emailOrUsername);

if ($user && Security::verifyPassword($password, $user['password'])) {
$_SESSION['user_id'] = $user['id'];
$_SESSION['username'] = $user['username'];
$_SESSION['first_name'] = $user['first_name'];
$_SESSION['last_name'] = $user['last_name'];
$_SESSION['institution_id'] = $user['institution_id'];
$_SESSION['roles'] = $this->userModel->getUserRoles($user['id']);
$_SESSION['is_superadmin'] = !empty($user['is_superadmin']);

```

```
header('Location: ' . BASE_URL . '/public/index.php?action=dashboard');
exit;
} else {
$error = 'Credenciales incorrectas';
}
}

include BASE_PATH . '/views/auth/login.php';
}

public function logout() {
session_destroy();
header('Location: ' . BASE_URL . '/public/index.php?action=login');
exit;
}

public function forgotPassword() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$email = Security::sanitize($_POST['email']);
$user = $this->userModel->findByEmail($email);

if ($user) {
$token = bin2hex(random_bytes(32));
$this->userModel->setResetToken($email, $token);

$resetLink = BASE_URL . "/public/index.php?action=reset&token={$token}";
$body = "Haga clic aquí para restablecer su contraseña: <a href='{$resetLink}'>{$resetLink}</a>";

Mailer::send($email, 'Restablecer contraseña', $body);
}

$message = 'Si el correo existe, recibirá un enlace de recuperación';
}

include BASE_PATH . '/views/auth/forgot.php';
}

public function resetPassword() {
$token = $_GET['token'] ?? '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$user = $this->userModel->validateResetToken($token);

if ($user && $_POST['password'] === $_POST['password_confirm']) {
$this->userModel->resetPassword($user['id'], $_POST['password']);
header('Location: ' . BASE_URL . '/public/index.php?action=login&reset=1');
exit;
}
}

include BASE_PATH . '/views/auth/reset.php';
}
}
```

4.5 — BackupController.php

controllers/BackupController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/helpers/Backup.php';

class BackupController {
private $backup;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$this->backup = new Backup();
}

public function index() {
$backups = $this->backup->getBackups();
include BASE_PATH . '/views/backup/index.php';
}

public function create() {
$filename = $this->backup->createBackup();

if ($filename) {
header('Location: ?action=backups&success=1&file=' . $filename);
} else {
header('Location: ?action=backups&error=1');
}
exit;
}

public function download() {
$filename = $_GET['file'] ?? '';
$this->backup->downloadBackup($filename);
}

public function cleanup() {
$deleted = $this->backup->deleteOldBackups(30);
header('Location: ?action=backups&cleanup=' . $deleted);
exit;
}

public function delete() {
$filename = $_GET['file'] ?? '';

if ($this->backup->deleteBackup($filename)) {
header('Location: ?action=backups&deleted=1');
} else {
header('Location: ?action=backups&error_delete=1');
}
exit;
}
}
```

4.6 — DashboardController.php

controllers/DashboardController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/Course.php';

class DashboardController {
private $attendanceModel;
private $userModel;
private $courseModel;

public function __construct() {
Security::requireLogin();

$db = new Database();
$this->attendanceModel = new Attendance($db);
$this->userModel = new User($db);
$this->courseModel = new Course($db);
}

public function index() {
$stats = $this->getStats();
include BASE_PATH . '/views/dashboard/index.php';
}

private function getStats() {
$stats = [];

if (Security::hasRole('autoridad')) {
$stats['total_students'] = $this->getTotalByRole('estudiante');
$stats['total_teachers'] = $this->getTotalByRole('docente');
$stats['total_courses'] = $this->getTotalCourses();
$stats['today_attendance'] = $this->getTodayAttendanceStats();
}

if (Security::hasRole('docente')) {
$stats['my_courses'] = $this->getTeacherCourses($_SESSION['user_id']);
$stats['my_students'] = $this->getTeacherStudentsCount($_SESSION['user_id']);
$stats['tutor_course'] = $this->getTutorCourse($_SESSION['user_id']);
}

if (Security::hasRole('estudiante')) {
$stats['my_attendance'] = $this->getStudentAttendanceStats($_SESSION['user_id']);
}

if (Security::hasRole('representante')) {
$stats['my_children'] = $this->getRepresentativeChildrenCount($_SESSION['user_id']);
}

return $stats;
}

private function getTotalByRole($role) {
$sql = "SELECT COUNT(DISTINCT u.id) as total
FROM users u
INNER JOIN user_roles ur ON u.id = ur.user_id
INNER JOIN roles r ON ur.role_id = r.id
WHERE r.name = :role AND u.institution_id = :institution_id";

$db = new Database();
```

```
$stmt = $db->connect()->prepare($sql);
$stmt->execute([
':role' => $role,
':institution_id' => $_SESSION['institution_id']
]);

return $stmt->fetch()['total'];
}

private function getTotalCourses() {
$sql = "SELECT COUNT(*) as total FROM courses WHERE institution_id = :institution_id";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':institution_id' => $_SESSION['institution_id']]);

return $stmt->fetch()['total'];
}

private function getTodayAttendanceStats() {
$sql = "SELECT
COUNT(*) as total,
SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN status = 'ausente' THEN 1 ELSE 0 END) as ausente
FROM attendances
WHERE date = CURDATE()";

$db = new Database();
$stmt = $db->connect()->query($sql);
return $stmt->fetch();
}

private function getTeacherCourses($teacherId) {
$sql = "SELECT COUNT(DISTINCT course_id) as total
FROM teacher_assignments
WHERE teacher_id = :teacher_id";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':teacher_id' => $teacherId]);

return $stmt->fetch()['total'];
}

private function getTeacherStudentsCount($teacherId) {
$sql = "SELECT COUNT(DISTINCT cs.student_id) as total
FROM teacher_assignments ta
INNER JOIN course_students cs ON ta.course_id = cs.course_id
WHERE ta.teacher_id = :teacher_id";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':teacher_id' => $teacherId]);

return $stmt->fetch()['total'];
}

private function getStudentAttendanceStats($studentId) {
$sql = "SELECT
COUNT(*) as total,
```

```
SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN status = 'ausente' THEN 1 ELSE 0 END) as ausente,
SUM(CASE WHEN status = 'tardanza' THEN 1 ELSE 0 END) as tardanza
FROM attendances
WHERE student_id = :student_id
AND date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':student_id' => $studentId]);

return $stmt->fetch();
}

private function getRepresentativeChildrenCount($representativeId) {
$sql = "SELECT COUNT(*) as total
FROM representatives
WHERE representative_id = :representative_id";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':representative_id' => $representativeId]);

return $stmt->fetch()['total'];
}

private function getTutorCourse($teacherId) {
$sql = "SELECT c.name as course_name
FROM teacher_assignments ta
INNER JOIN courses c ON ta.course_id = c.id
INNER JOIN school_years sy ON ta.school_year_id = sy.id
WHERE ta.teacher_id = :teacher_id
AND ta.is_tutor = 1
AND sy.is_active = 1
LIMIT 1";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':teacher_id' => $teacherId]);
$row = $stmt->fetch();
return $row ? $row['course_name'] : null;
}
}
```

4.7 — ExportController.php

controllers/ExportController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/vendor/autoload.php';
require_once BASE_PATH . '/models/Course.php';

use PhpOffice\PhpSpreadsheet\Spreadsheet;
use PhpOffice\PhpSpreadsheet\Writer\Xlsx;
use PhpOffice\PhpSpreadsheet\Style\Fill;
use PhpOffice\PhpSpreadsheet\Style\Alignment;

class ExportController {
private $courseModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole(['autoridad', 'docente'])) {
die('Acceso denegado');
}

$db = new Database();
$this->courseModel = new Course($db);
}

public function exportStudentList() {
$courseId = (int)$_GET['course_id'];
$students = $this->courseModel->getEnrolledStudents($courseId);

$courses = $this->courseModel->getAll();
$course = array_filter($courses, fn($c) => $c['id'] == $courseId);
$course = reset($course);

$spreadsheet = new Spreadsheet();
$sheet = $spreadsheet->getActiveSheet();

$sheet->setTitle('Nómina');

// Encabezado
$sheet->mergeCells('A1:E1');
$sheet->setCellValue('A1', 'NÓMINA DE ESTUDIANTES');
$sheet->getStyle('A1')->getFont()->setBold(true)->setSize(16);
$sheet->getStyle('A1')->getAlignment()->setHorizontal(Alignment::HORIZONTAL_CENTER);

$sheet->setCellValue('A3', 'Curso:');
$sheet->setCellValue('B3', $course['name'] . ' - ' . $course['shift_name']);
$sheet->getStyle('A3')->getFont()->setBold(true);

$sheet->setCellValue('A4', 'Total:');
$sheet->setCellValue('B4', count($students) . ' estudiantes');
$sheet->getStyle('A4')->getFont()->setBold(true);

// Cabecera tabla
$headers = ['#', 'Apellidos', 'Nombres', 'Cédula', 'Fecha Matrícula'];
$col = 'A';
foreach ($headers as $header) {
$sheet->setCellValue($col . '6', $header);
$sheet->getStyle($col . '6')->getFont()->setBold(true);
$sheet->getStyle($col . '6')->getFill()
->setFillType(Fill::FILL_SOLID)
->getStartColor()->setRGB('007bff');
$sheet->getStyle($col . '6')->getFont()->getColor()->setRGB('FFFFFF');
```

```
$col++;
}

// Datos
$row = 7;
$num = 1;
foreach ($students as $student) {
$sheet->setCellValue('A' . $row, $num++);
$sheet->setCellValue('B' . $row, $student['last_name']);
$sheet->setCellValue('C' . $row, $student['first_name']);
$sheet->setCellValue('D' . $row, $student['dni'] ?? '-');
$sheet->setCellValue('E' . $row, date('d/m/Y', strtotime($student['enrollment_date'])));
$row++;
}

// Ajustar anchos
$sheet->getColumnDimension('A')->setWidth(5);
$sheet->getColumnDimension('B')->setWidth(20);
$sheet->getColumnDimension('C')->setWidth(20);
$sheet->getColumnDimension('D')->setWidth(15);
$sheet->getColumnDimension('E')->setWidth(15);

$writer = new Xlsx($spreadsheet);

header('Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');
header('Content-Disposition: attachment;filename="nomina_' . $course['name'] . '_' . date('YmdHis') .
'.xlsx"');
header('Cache-Control: max-age=0');

$writer->save('php://output');
exit;
}
}
```

4.8 — InstitutionController.php

controllers/InstitutionController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Institution.php';
require_once BASE_PATH . '/models/InstitutionShift.php';
require_once BASE_PATH . '/models/Shift.php';

class InstitutionController {
private $institutionModel;
private $institutionShiftModel;
private $shiftModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$db = new Database();
$this->institutionModel = new Institution($db);
$this->institutionShiftModel = new InstitutionShift($db);
$this->shiftModel = new Shift($db);
}

public function index() {
$institution = $this->institutionModel->getById(1);
$allShifts = $this->shiftModel->getAll();
$assignedShiftIds = $this->institutionShiftModel->getInstitutionShiftIds(1);

include BASE_PATH . '/views/institution/index.php';
}

public function update() {
if ($_SERVER['REQUEST_METHOD'] !== 'POST') return;

$current = $this->institutionModel->getById(1);
$logoPath = $current['logo_path']; // Mantener logo actual por defecto

if (isset($_FILES['logo']) && $_FILES['logo']['error'] == 0) {
$uploadDir = BASE_PATH . '/uploads/institution/';
if (!file_exists($uploadDir)) {
mkdir($uploadDir, 0755, true);
}

$ext = strtolower(pathinfo($_FILES['logo']['name'], PATHINFO_EXTENSION));
$allowed = ['jpg', 'jpeg', 'png', 'gif', 'webp'];
$maxSize = 2 * 1024 * 1024; // 2MB

if (in_array($ext, $allowed) && $_FILES['logo']['size'] <= $maxSize) {
$filename = 'logo_' . time() . '.' . $ext;
$uploadPath = $uploadDir . $filename;

if (move_uploaded_file($_FILES['logo']['tmp_name'], $uploadPath)) {
// Eliminar logo anterior
if ($current['logo_path'] && file_exists(BASE_PATH . '/' . $current['logo_path'])) {
unlink(BASE_PATH . '/' . $current['logo_path']);
}
$logoPath = 'uploads/institution/' . $filename;
}
}
}
}
```

```
61
62 $data = [
63 'name' => Security::sanitize($_POST['name']),
64 'address' => Security::sanitize($_POST['address']),
65 'province' => Security::sanitize($_POST['province']),
66 'city' => Security::sanitize($_POST['city']),
67 'phone' => Security::sanitize($_POST['phone']),
68 'email' => Security::sanitize($_POST['email']),
69 'director_name'=> Security::sanitize($_POST['director_name']),
70 'amie_code' => Security::sanitize($_POST['amie_code']),
71 'website' => Security::sanitize($_POST['website']),
72 'logo_path' => $logoPath,
73 ];
74
75 $this->institutionModel->update(1, $data);
76 header('Location: ?action=institution&success=1');
77 exit;
78 }
79
80 // Toggle: si está asignada la quita, si no la asigna — responde JSON para AJAX
81 public function toggleShift() {
82 header('Content-Type: application/json');
83
84 if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
85 echo json_encode(['error' => 'Método no permitido']); exit;
86 }
87
88 $shiftId = (int)$_POST['shift_id'];
89 $assignedShiftIds = $this->institutionShiftModel->getInstitutionShiftIds(1);
90
91 if (in_array($shiftId, $assignedShiftIds)) {
92 $this->institutionShiftModel->remove(1, $shiftId);
93 echo json_encode(['action' => 'removed', 'shift_id' => $shiftId]);
94 } else {
95 $this->institutionShiftModel->assign([
96 ':institution_id' => 1,
97 ':shift_id' => $shiftId,
98 ]);
99 echo json_encode(['action' => 'assigned', 'shift_id' => $shiftId]);
100 }
101 exit;
102 }
103
104 // Mantener compatibilidad con rutas antiguas
105 public function assignShift() {
106 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
107 $this->institutionShiftModel->assign([
108 ':institution_id' => 1,
109 ':shift_id' => (int)$_POST['shift_id'],
110 ]);
111 header('Location: ?action=institution&success=1');
112 exit;
113 }
114 }
115
116 public function removeShift() {
117 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
118 $this->institutionShiftModel->remove(1, (int)$_POST['shift_id']);
119 header('Location: ?action=institution&success=1');
120 exit;
```

```
121 }
122 }
123 }
```

4.9 — JustificationController.php

controllers/JustificationController.php

PHP

```php
<?php
// ARCHIVO COMPLETO - Reemplaza TODO el archivo
// controllers/JustificationController.php

require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Justification.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/Notification.php';

class JustificationController {
private $justificationModel;
private $attendanceModel;
private $notificationModel;

public function __construct() {
Security::requireLogin();

$db = new Database();
$this->justificationModel = new Justification($db);
$this->attendanceModel = new Attendance($db);
$this->notificationModel = new Notification($db);
}

public function submit() {
if (!Security::hasRole(['estudiante', 'representante'])) {
die('Acceso denegado');
}

$studentId = Security::hasRole('estudiante')
? $_SESSION['user_id']
: (isset($_GET['student_id']) ? (int)$_GET['student_id'] : null);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

$studentId = Security::hasRole('estudiante')
? $_SESSION['user_id']
: (int)$_POST['student_id'];

// Ausencias seleccionadas
$attendanceIds = $_POST['attendance_ids'] ?? [];
if (empty($attendanceIds)) {
header('Location: ?action=submit_justification&error=no_absences');
exit;
}

// Obtener fechas de las ausencias seleccionadas para calcular días laborables
$db = new Database();
$pdo = $db->connect();
$in = implode(',', array_map('intval', $attendanceIds));
$rows = $pdo->query(
"SELECT date FROM attendances WHERE id IN ($in) AND student_id = $studentId ORDER BY date"
)->fetchAll();

if (empty($rows)) {
header('Location: ?action=submit_justification&error=no_absences');
exit;
}

$dates = array_column($rows, 'date');
```

```
$dateFrom = min($dates);
$dateTo = max($dates);
$workingDays = count($attendanceIds); // cada ausencia = 1 día (pueden ser no consecutivos)
$canApprove = Justification::resolveApprover($workingDays);

// Subir documento
$documentPath = null;
if (isset($_FILES['document']) && $_FILES['document']['error'] == 0) {
$uploadDir = BASE_PATH . '/uploads/justifications/';
if (!file_exists($uploadDir)) mkdir($uploadDir, 0755, true);
$ext = strtolower(pathinfo($_FILES['document']['name'], PATHINFO_EXTENSION));
$allowed = ['pdf','jpg','jpeg','png'];
if (in_array($ext, $allowed) && $_FILES['document']['size'] <= 5 * 1024 * 1024) {
$filename = uniqid() . '.' . $ext;
if (move_uploaded_file($_FILES['document']['tmp_name'], $uploadDir . $filename)) {
$documentPath = 'uploads/justifications/' . $filename;
}
}
}

// Motivo
$reasonType = Security::sanitize($_POST['reason_type'] ?? '');
$reasonText = Security::sanitize($_POST['reason'] ?? '');
$reasonFinal = ($reasonType === 'Otro')
? $reasonText
: ($reasonType . ($reasonText ? ': ' . $reasonText : ''));

$data = [
':student_id' => $studentId,
':submitted_by' => $_SESSION['user_id'],
':date_from' => $dateFrom,
':date_to' => $dateTo,
':working_days' => $workingDays,
':reason_type' => $reasonType,
':reason' => $reasonFinal,
':document_path' => $documentPath,
':can_approve' => $canApprove,
];

$this->justificationModel->createForAttendances($attendanceIds, $data);

// Notificar
if ($canApprove === 'tutor') {
$stmt = $pdo->prepare(
"SELECT ta.teacher_id FROM teacher_assignments ta
INNER JOIN course_students cs ON ta.course_id = cs.course_id
WHERE cs.student_id = :sid AND ta.is_tutor = 1 LIMIT 1"
);
$stmt->execute([':sid' => $studentId]);
$tutorId = $stmt->fetchColumn();
if ($tutorId) {
$this->notificationModel->create(
$tutorId,
'■ Justificación pendiente (tutor)',
"Un estudiante de tu curso necesita justificación por $workingDays día(s).",
'info',
'?action=tutor_pending_justifications'
);
}
} else {
```

```
$this->_notifyReviewers(
'■ Nueva justificación pendiente',
"Justificación de $workingDays día(s) requiere revisión.",
'info',
'?action=pending_justifications'
);
}

header('Location: ?action=my_justifications&success=1');
exit;
}

// GET: cargar ausencias no justificadas
$absences = $this->attendanceModel->getUnjustifiedAbsences($studentId ?? $_SESSION['user_id']);

// Si representante, cargar sus hijos
$myChildren = [];
if (Security::hasRole('representante')) {
$db = new Database();
$stmt = $db->connect()->prepare(
"SELECT u.id, CONCAT(u.last_name,' ',u.first_name) as full_name
FROM users u INNER JOIN representatives r ON u.id = r.student_id
WHERE r.representative_id = :rid ORDER BY u.last_name"
);
$stmt->execute([':rid' => $_SESSION['user_id']]);
$myChildren = $stmt->fetchAll();
}

include BASE_PATH . '/views/justifications/submit.php';
}


public function pendingForTutor() {
if (!Security::hasRole('docente')) die('Acceso denegado');

$db = new Database();
$attModel = new Attendance($db);
$courseId = $attModel->getTutorCourseId($_SESSION['user_id']);

if (!$courseId) {
header('Location: ?action=dashboard&error=not_tutor'); exit;
}

$justifications = $this->justificationModel->getPendingForTutor($courseId);
include BASE_PATH . '/views/justifications/pending.php';
}

public function myJustifications() {
if (!Security::hasRole(['estudiante', 'representante'])) {
die('Acceso denegado');
}
$studentId = $_SESSION['user_id'];
$justifications = $this->justificationModel->getByStudent($studentId);
include BASE_PATH . '/views/justifications/my_list.php';
}

public function pending() {
if (!Security::hasRole(['autoridad', 'inspector'])) {
die('Acceso denegado');
}
```

```
$justifications = $this->justificationModel->getPending();
include BASE_PATH . '/views/justifications/pending.php';
}

public function review() {
if (!Security::hasRole(['autoridad', 'inspector'])) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$justificationId = (int)$_POST['justification_id'];
$action = $_POST['review_action'];
$notes = Security::sanitize($_POST['notes'] ?? '');

// Obtener datos ANTES de actualizar para poder notificar
$justification = $this->justificationModel->getById($justificationId);

if ($action === 'approve') {
$this->justificationModel->approveRange($justificationId, $_SESSION['user_id'], $notes);

if ($justification) {
// Notificar al estudiante — enlace a sus justificaciones
$this->notificationModel->create(
$justification['student_id'],
'■ Justificación aprobada',
'Tu justificación fue aprobada.' . ($notes ? " Nota: $notes" : ''),
'success',
'?action=my_justifications'
);
// Notificar al representante si fue distinto del estudiante
if ($justification['submitted_by'] != $justification['student_id']) {
$this->notificationModel->create(
$justification['submitted_by'],
'■ Justificación aprobada',
'La justificación que enviaste fue aprobada.' . ($notes ? " Nota: $notes" : ''),
'success',
'?action=my_justifications'
);
}
}
} else {
$this->justificationModel->reject($justificationId, $_SESSION['user_id'], $notes);

if ($justification) {
// Notificar al estudiante — puede enviar una nueva justificación
$this->notificationModel->create(
$justification['student_id'],
'■ Justificación rechazada',
'Tu justificación fue rechazada.' . ($notes ? " Motivo: $notes" : ''),
'danger',
'?action=my_justifications'
);
if ($justification['submitted_by'] != $justification['student_id']) {
$this->notificationModel->create(
$justification['submitted_by'],
'■ Justificación rechazada',
'La justificación que enviaste fue rechazada.' . ($notes ? " Motivo: $notes" : ''),
'danger',
'?action=my_justifications'
);
```

```
}
}
}

header('Location: ?action=pending_justifications&success=1');
exit;
}
}

public function reviewed() {
if (!Security::hasRole(['autoridad', 'inspector'])) {
die('Acceso denegado');
}
$filter = $_GET['filter'] ?? 'all';
$justifications = $this->justificationModel->getReviewed();

if ($filter !== 'all') {
$justifications = array_values(array_filter($justifications, function($j) use ($filter) {
return $j['status'] === $filter;
}));
}

include BASE_PATH . '/views/justifications/reviewed.php';
}

// Privado: notificar a todos los revisores
private function _notifyReviewers($title, $message, $type = 'info', $link = null) {
$db = new Database();
$pdo = $db->connect();

$sql = "SELECT DISTINCT u.id FROM users u
INNER JOIN user_roles ur ON u.id = ur.user_id
INNER JOIN roles r ON ur.role_id = r.id
WHERE r.name IN ('autoridad','inspector') AND u.id != :sender";
$stmt = $pdo->prepare($sql);
$stmt->execute([':sender' => $_SESSION['user_id']]);
$reviewers = $stmt->fetchAll(PDO::FETCH_COLUMN);

foreach ($reviewers as $reviewerId) {
$this->notificationModel->create($reviewerId, $title, $message, $type, $link);
}
}
}
```

4.10 — NotificationController.php

controllers/NotificationController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Notification.php';

class NotificationController {
private $notificationModel;

public function __construct() {
Security::requireLogin();
$db = new Database();
$this->notificationModel = new Notification($db);
}

// Vista principal: todas las notificaciones del usuario
public function index() {
$userId = $_SESSION['user_id'];
$perPage = 15;
$page = max(1, (int)($_GET['page'] ?? 1));
$offset = ($page - 1) * $perPage;

$notifications = $this->notificationModel->getByUser($userId, $perPage, $offset);
$total = $this->notificationModel->countByUser($userId);
$totalPages = (int)ceil($total / $perPage);
$unreadCount = $this->notificationModel->getUnreadCount($userId);

include BASE_PATH . '/views/notifications/index.php';
}

// AJAX: marcar una como leída y devolver nuevo conteo
public function markRead() {
$userId = $_SESSION['user_id'];
$id = (int)($_GET['id'] ?? 0);

if ($id) {
$this->notificationModel->markAsRead($id, $userId);
}

// Responder JSON para el dropdown del navbar
header('Content-Type: application/json');
echo json_encode([
'success' => true,
'unread' => $this->notificationModel->getUnreadCount($userId)
]);
exit;
}

// Marcar todas como leídas (POST desde vista)
public function markAllRead() {
$userId = $_SESSION['user_id'];
$this->notificationModel->markAllAsRead($userId);
header('Location: ?action=notifications&success=read');
exit;
}

// Eliminar una notificación (POST desde vista)
public function delete() {
$userId = $_SESSION['user_id'];
$id = (int)($_POST['notification_id'] ?? 0);

if ($id) {
```

```php
$this->notificationModel->delete($id, $userId);
}

header('Location: ?action=notifications&success=deleted');
exit;
}

// Eliminar todas las leídas
public function deleteRead() {
$userId = $_SESSION['user_id'];
$this->notificationModel->deleteRead($userId);
header('Location: ?action=notifications&success=cleaned');
exit;
}

// AJAX: obtener no leídas para el dropdown del navbar
public function getUnread() {
$userId = $_SESSION['user_id'];
$notifications = $this->notificationModel->getUnread($userId, 8);
$count = $this->notificationModel->getUnreadCount($userId);

header('Content-Type: application/json');
echo json_encode([
'notifications' => $notifications,
'count' => $count
]);
exit;
}
}
```

4.11 — ProfileController.php

controllers/ProfileController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/User.php';

class ProfileController {
private $userModel;

public function __construct() {
Security::requireLogin();

$db = new Database();
$this->userModel = new User($db);
}

public function view() {
$user = $this->userModel->findById($_SESSION['user_id']);
$roles = $this->userModel->getUserRoles($_SESSION['user_id']);

include BASE_PATH . '/views/profile/view.php';
}

public function edit() {
$user = $this->userModel->findById($_SESSION['user_id']);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$data = [
'first_name' => Security::sanitize($_POST['first_name']),
'last_name' => Security::sanitize($_POST['last_name']),
'phone' => Security::sanitize($_POST['phone']),
'email' => Security::sanitize($_POST['email'])
];

if ($this->userModel->updateProfile($_SESSION['user_id'], $data)) {
header('Location: ?action=profile&success=1');
exit;
}
}

include BASE_PATH . '/views/profile/edit.php';
}

public function changePassword() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$currentPassword = $_POST['current_password'];
$newPassword = $_POST['new_password'];
$confirmPassword = $_POST['confirm_password'];

$user = $this->userModel->findById($_SESSION['user_id']);

if (!Security::verifyPassword($currentPassword, $user['password'])) {
$error = 'Contraseña actual incorrecta';
} elseif ($newPassword !== $confirmPassword) {
$error = 'Las contraseñas no coinciden';
} elseif (strlen($newPassword) < 6) {
$error = 'La contraseña debe tener al menos 6 caracteres';
} else {
if ($this->userModel->updatePassword($_SESSION['user_id'], $newPassword)) {
header('Location: ?action=profile&password_changed=1');
exit;
}
}

include BASE_PATH . '/views/profile/change_password.php';
}
}
```

4.12 — ReportController.php

controllers/ReportController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/Subject.php';
require_once BASE_PATH . '/models/SchoolYear.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/Institution.php';
require_once BASE_PATH . '/vendor/autoload.php';

use PhpOffice\PhpSpreadsheet\Spreadsheet;
use PhpOffice\PhpSpreadsheet\Writer\Xlsx;
use PhpOffice\PhpSpreadsheet\Style\Fill;
use PhpOffice\PhpSpreadsheet\Style\Border;
use PhpOffice\PhpSpreadsheet\Style\Alignment;

class ReportController {
private $attendanceModel;
private $courseModel;
private $subjectModel;
private $schoolYearModel;
private $userModel;
private $institutionModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole(['autoridad', 'inspector', 'docente'])) {
die('Acceso denegado');
}

$db = new Database();
$this->attendanceModel = new Attendance($db);
$this->courseModel = new Course($db);
$this->subjectModel = new Subject($db);
$this->schoolYearModel = new SchoolYear($db);
$this->userModel = new User($db);
$this->institutionModel = new Institution($db);
}

public function index() {
$courses = $this->courseModel->getAll();
$subjects = $this->subjectModel->getAll();

$data = [];
$course = null;
$startDate = null;
$endDate = null;

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['preview'])) {
$courseId = (int)$_POST['course_id'];
$startDate = $_POST['start_date'];
$endDate = $_POST['end_date'];

$data = $this->attendanceModel->getReportData($courseId, $startDate, $endDate);
$course = $this->getCourseInfo($courseId);
}

include BASE_PATH . '/views/reports/index.php';
}

```

```
public function generatePDF() {
$courseId = (int)$_POST['course_id'];
$startDate = $_POST['start_date'];
$endDate = $_POST['end_date'];

$data = $this->attendanceModel->getReportData($courseId, $startDate, $endDate);
$course = $this->getCourseInfo($courseId);
$institution = $this->institutionModel->getById($_SESSION['institution_id']);

$pdf = new \TCPDF('L', 'mm', 'A4', true, 'UTF-8');

$pdf->SetCreator('EcuAsist 2026');
$pdf->SetAuthor('Sistema de Asistencia');
$pdf->SetTitle('Reporte de Asistencia');

$pdf->setPrintHeader(false);
$pdf->setPrintFooter(false);
$pdf->SetMargins(10, 10, 10);
$pdf->SetAutoPageBreak(true, 10);

$pdf->AddPage();

$pdf->SetFont('helvetica', 'B', 16);
$pdf->Cell(0, 10, 'REPORTE DE ASISTENCIA', 0, 1, 'C');

$pdf->SetFont('helvetica', '', 10);
$pdf->Cell(0, 5, $institution['name'] ?? 'Institución Educativa', 0, 1, 'C');
$pdf->Ln(5);

$pdf->SetFont('helvetica', 'B', 11);
$pdf->Cell(50, 6, 'Curso:', 0, 0);
$pdf->SetFont('helvetica', '', 11);
$pdf->Cell(0, 6, html_entity_decode($course['name'], ENT_QUOTES, 'UTF-8'), 0, 1);

$pdf->SetFont('helvetica', 'B', 11);
$pdf->Cell(50, 6, 'Período:', 0, 0);
$pdf->SetFont('helvetica', '', 11);
$pdf->Cell(0, 6, date('d/m/Y', strtotime($startDate)) . ' al ' . date('d/m/Y', strtotime($endDate)), 0, 1);
$pdf->Ln(3);

$html = '<table border="1" cellpadding="4">
<thead>
<tr style="background-color: #007bff; color: white; font-weight: bold;">
<th width="16.666666%">Fecha</th>
<th width="16.66666%">Estudiante</th>
<th width="16.66666%">Asignatura</th>
<th width="16.66666%">Hora</th>
<th width="16.66666%">Estado</th>
<th width="16.66666%">Observación</th>
</tr>
</thead>
<tbody>';

foreach ($data as $row) {
$statusColor = $this->getStatusColor($row['status']);
$html .= '<tr>
<td>' . date('d/m/Y', strtotime($row['date'])) . '</td>
<td>' . $row['student_name'] . '</td>
<td>' . $row['subject_name'] . '</td>
<td>' . $row['hour_period'] . '</td>
```

```
<td style="background-color: ' . $statusColor . ';">' . ucfirst($row['status']) . '</td>
<td>' . ($row['observation'] ?: '-') . '</td>
</tr>';
}

$html .= '</tbody></table>';

$pdf->SetFont('helvetica','',9);

$pdf->writeHTML($html, true, false, true, false, '');

// Generar nombre de archivo con curso
$courseName = $this->sanitizeFilename($course['name']);
$filename = 'reporte_asistencia_' . $courseName . '_' . date('YmdHis') . '.pdf';

$pdf->Output($filename, 'D');
exit;
}

public function generateExcel() {
$courseId = (int)$_POST['course_id'];
$startDate = $_POST['start_date'];
$endDate = $_POST['end_date'];

$data = $this->attendanceModel->getReportData($courseId, $startDate, $endDate);
$course = $this->getCourseInfo($courseId);
$institution = $this->institutionModel->getById($_SESSION['institution_id']);

$spreadsheet = new Spreadsheet();
$sheet = $spreadsheet->getActiveSheet();

$sheet->setTitle('Asistencia');

// Encabezado
$sheet->mergeCells('A1:F1');
$sheet->setCellValue('A1', 'REPORTE DE ASISTENCIA');
$sheet->getStyle('A1')->getFont()->setBold(true)->setSize(16);
$sheet->getStyle('A1')->getAlignment()->setHorizontal(Alignment::HORIZONTAL_CENTER);

$sheet->mergeCells('A2:F2');
$sheet->setCellValue('A2', $institution['name'] ?? 'Institución Educativa');
$sheet->getStyle('A2')->getAlignment()->setHorizontal(Alignment::HORIZONTAL_CENTER);

$sheet->setCellValue('A4', 'Curso:');
$sheet->setCellValue('B4', html_entity_decode($course['name'], ENT_QUOTES, 'UTF-8'));
$sheet->getStyle('A4')->getFont()->setBold(true);

$sheet->setCellValue('A5', 'Período:');
$sheet->setCellValue('B5', date('d/m/Y', strtotime($startDate)) . ' al ' . date('d/m/Y',
strtotime($endDate)));
$sheet->getStyle('A5')->getFont()->setBold(true);

// Cabecera de tabla
$row = 7;
$headers = ['Fecha', 'Estudiante', 'Asignatura', 'Hora', 'Estado', 'Observación'];
$col = 'A';

foreach ($headers as $header) {
$sheet->setCellValue($col . $row, $header);
$sheet->getStyle($col . $row)->getFont()->setBold(true);
$sheet->getStyle($col . $row)->getFill()
```

```
->setFillType(Fill::FILL_SOLID)
->getStartColor()->setRGB('007bff');
$sheet->getStyle($col . $row)->getFont()->getColor()->setRGB('FFFFFF');
$col++;
}

// Datos
$row = 8;
foreach ($data as $record) {
$sheet->setCellValue('A' . $row, date('d/m/Y', strtotime($record['date'])));
$sheet->setCellValue('B' . $row, $record['student_name']);
$sheet->setCellValue('C' . $row, $record['subject_name']);
$sheet->setCellValue('D' . $row, $record['hour_period']);
$sheet->setCellValue('E' . $row, ucfirst($record['status']));
$sheet->setCellValue('F' . $row, $record['observation'] ?: '-');

$statusColor = $this->getStatusColorHex($record['status']);
$sheet->getStyle('E' . $row)->getFill()
->setFillType(Fill::FILL_SOLID)
->getStartColor()->setRGB($statusColor);

$row++;
}

// Ajustar anchos
$sheet->getColumnDimension('A')->setWidth(12);
$sheet->getColumnDimension('B')->setWidth(25);
$sheet->getColumnDimension('C')->setWidth(20);
$sheet->getColumnDimension('D')->setWidth(15);
$sheet->getColumnDimension('E')->setWidth(12);
$sheet->getColumnDimension('F')->setWidth(25);

// Bordes
$styleArray = [
'borders' => [
'allBorders' => [
'borderStyle' => Border::BORDER_THIN,
],
],
];
$sheet->getStyle('A7:F' . ($row - 1))->applyFromArray($styleArray);

$writer = new Xlsx($spreadsheet);

// Generar nombre de archivo con curso
$courseName = $this->sanitizeFilename($course['name']);
$filename = 'reporte_asistencia_' . $courseName . '_' . date('YmdHis') . '.xlsx';

header('Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');
header('Content-Disposition: attachment;filename="' . $filename . '"');
header('Cache-Control: max-age=0');

$writer->save('php://output');
exit;
}

private function getCourseInfo($courseId) {
$courses = $this->courseModel->getAll();
foreach ($courses as $course) {
if ($course['id'] == $courseId) {
```

```
return $course;
}
}
return null;
}

private function getStatusColor($status) {
switch ($status) {
case 'presente': return '#d4edda';
case 'ausente': return '#f8d7da';
case 'tardanza': return '#fff3cd';
case 'justificado': return '#d1ecf1';
default: return '#ffffff';
}
}

private function getStatusColorHex($status) {
switch ($status) {
case 'presente': return '28a745';
case 'ausente': return 'dc3545';
case 'tardanza': return 'ffc107';
case 'justificado': return '17a2b8';
default: return 'FFFFFF';
}
}

private function sanitizeFilename($filename) {
// Decodificar HTML entities
$filename = html_entity_decode($filename, ENT_QUOTES, 'UTF-8');

// Reemplazar caracteres no permitidos en nombres de archivo
$filename = str_replace(['"', '/', '\\', ':', '*', '?', '<', '>', '|'], '', $filename);

// Reemplazar espacios con guiones bajos
$filename = str_replace(' ', '_', $filename);

// Reemplazar guiones múltiples
$filename = preg_replace('/-+/', '-', $filename);

// Limitar longitud (máximo 50 caracteres)
if (strlen($filename) > 50) {
$filename = substr($filename, 0, 50);
}

// Limpiar caracteres al inicio y final
$filename = trim($filename, '-_');

return $filename;
}
}
```

4.13 — RepresentativeController.php

controllers/RepresentativeController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Representative.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/Attendance.php';

class RepresentativeController {
private $representativeModel;
private $userModel;
private $attendanceModel;

public function __construct() {
Security::requireLogin();

$db = new Database();
$this->representativeModel = new Representative($db);
$this->userModel = new User($db);
$this->attendanceModel = new Attendance($db);
}

public function manageRepresentatives() {
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$representatives = $this->userModel->getByRole('representante');
$students = $this->userModel->getByRole('estudiante');

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$repId = (int)$_POST['representative_id'];
$studentId = (int)$_POST['student_id'];
$relationship = Security::sanitize($_POST['relationship']);
$isPrimary = isset($_POST['is_primary']) ? 1 : 0;

$this->representativeModel->assignStudent($repId, $studentId, $relationship, $isPrimary);
header('Location: ?action=manage_representatives&success=1');
exit;
}

include BASE_PATH . '/views/representatives/manage.php';
}

public function myChildren() {
if (!Security::hasRole('representante')) {
die('Acceso denegado');
}

$children = $this->representativeModel->getStudentsByRepresentative($_SESSION['user_id']);

include BASE_PATH . '/views/representatives/my_children.php';
}

public function childAttendance() {
if (!Security::hasRole('representante')) {
die('Acceso denegado');
}

$studentId = (int)($_GET['student_id'] ?? 0);

// Verificar que el estudiante sea hijo del representante
```

```php
$children = $this->representativeModel->getStudentsByRepresentative($_SESSION['user_id']);
$authorized = false;
$student = null;

foreach ($children as $child) {
if ($child['id'] == $studentId) {
$authorized = true;
$student = $child;
break;
}
}

if (!$authorized) {
die('No autorizado para ver este estudiante');
}

$attendances = $this->attendanceModel->getByStudent($studentId);

include BASE_PATH . '/views/representatives/child_attendance.php';
}

public function removeRelation() {
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$repId = (int)($_GET['rep_id'] ?? 0);
$studentId = (int)($_GET['student_id'] ?? 0);

if ($repId && $studentId) {
if ($this->representativeModel->removeStudent($repId, $studentId)) {
header('Location: ?action=manage_representatives&removed=1');
exit;
}
}

header('Location: ?action=manage_representatives&error=1');
exit;
}
}
```

4.14 — ScheduleController.php

controllers/ScheduleController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/ClassSchedule.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/Subject.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/SchoolYear.php';

class ScheduleController {
private $scheduleModel;
private $courseModel;
private $subjectModel;
private $userModel;
private $schoolYearModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$db = new Database();
$this->scheduleModel = new ClassSchedule($db);
$this->courseModel = new Course($db);
$this->subjectModel = new Subject($db);
$this->userModel = new User($db);
$this->schoolYearModel = new SchoolYear($db);
}

public function index() {
$courses = $this->courseModel->getAll();
include BASE_PATH . '/views/schedule/index.php';
}

public function manageCourse() {
$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Location: ?action=schedules');
exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$activeYear = $this->schoolYearModel->getActive();

$data = [
':course_id' => $courseId,
':subject_id' => (int)$_POST['subject_id'],
':teacher_id' => (int)$_POST['teacher_id'],
':school_year_id' => $activeYear['id'],
':day_of_week' => $_POST['day_of_week'],
':period_number' => (int)$_POST['period_number']
];

$result = $this->scheduleModel->create($data);

if ($result['success']) {
header('Location: ?action=manage_schedule&course_id=' . $courseId . '&success=1');
} else {
header('Location: ?action=manage_schedule&course_id=' . $courseId . '&error=' .
urlencode($result['message']));
```

```
}
exit;
}

$activeYear = $this->schoolYearModel->getActive();
$courses = $this->courseModel->getAll();
$course = array_filter($courses, fn($c) => $c['id'] == $courseId);
$course = reset($course);

$schedule = $this->scheduleModel->getByCourse($courseId, $activeYear['id']);
$subjects = $this->subjectModel->getAll();
$teachers = $this->userModel->getByRole('docente');

include BASE_PATH . '/views/schedule/manage.php';
}

public function deleteClass() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$scheduleId = (int)$_POST['schedule_id'];
$courseId = (int)$_POST['course_id'];

$this->scheduleModel->delete($scheduleId);
header('Location: ?action=manage_schedule&course_id=' . $courseId . '&deleted=1');
exit;
}
}

public function getCourseSubjectsSchedule() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

$courseId = (int)($_GET['course_id'] ?? 0);

if (!$courseId) {
header('Content-Type: application/json');
echo json_encode([]);
exit;
}

// Obtener asignaturas con sus docentes asignados al curso
$sql = "SELECT DISTINCT ta.subject_id, s.name as subject_name,
ta.teacher_id, CONCAT(u.last_name, ' ', u.first_name) as teacher_name
FROM teacher_assignments ta
INNER JOIN subjects s ON ta.subject_id = s.id
INNER JOIN users u ON ta.teacher_id = u.id
WHERE ta.course_id = :course_id
ORDER BY s.name";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':course_id' => $courseId]);
$subjects = $stmt->fetchAll(PDO::FETCH_ASSOC);

header('Content-Type: application/json');
echo json_encode($subjects);
exit;
}
```

```php

public function checkScheduleConflict() {
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['exists' => false]);
exit;
}

$courseId = (int)($_GET['course_id'] ?? 0);
$day = $_GET['day'] ?? '';
$period = (int)($_GET['period'] ?? 0);

if (!$courseId || !$day || !$period) {
header('Content-Type: application/json');
echo json_encode(['exists' => false]);
exit;
}

$db = new Database();
$activeYear = $this->schoolYearModel->getActive();

$sql = "SELECT s.name as subject_name, CONCAT(u.last_name, ' ', u.first_name) as teacher_name
FROM class_schedule cs
INNER JOIN subjects s ON cs.subject_id = s.id
INNER JOIN users u ON cs.teacher_id = u.id
WHERE cs.course_id = :course_id
AND cs.day_of_week = :day
AND cs.period_number = :period
AND cs.school_year_id = :school_year_id
LIMIT 1";

$stmt = $db->connect()->prepare($sql);
$stmt->execute([
':course_id' => $courseId,
':day' => $day,
':period' => $period,
':school_year_id' => $activeYear['id']
]);

$existing = $stmt->fetch(PDO::FETCH_ASSOC);

$result = [
'exists' => $existing ? true : false,
'subject_name' => $existing['subject_name'] ?? null,
'teacher_name' => $existing['teacher_name'] ?? null
];

header('Content-Type: application/json');
echo json_encode($result);
exit;
}
}
```

4.15 — SearchController.php

controllers/SearchController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/Attendance.php';

class SearchController {
private $userModel;
private $courseModel;
private $attendanceModel;

public function __construct() {
Security::requireLogin();

$db = new Database();
$this->userModel = new User($db);
$this->courseModel = new Course($db);
$this->attendanceModel = new Attendance($db);
}

public function searchStudents() {
$query = $_GET['q'] ?? '';

if (strlen($query) < 2) {
echo json_encode([]);
exit;
}

$sql = "SELECT u.id, u.first_name, u.last_name, u.dni, u.email,
c.name as course_name
FROM users u
INNER JOIN user_roles ur ON u.id = ur.user_id
INNER JOIN roles r ON ur.role_id = r.id
LEFT JOIN course_students cs ON u.id = cs.student_id
LEFT JOIN courses c ON cs.course_id = c.id
WHERE r.name = 'estudiante'
AND u.institution_id = :institution_id
AND (u.first_name LIKE :query
OR u.last_name LIKE :query
OR u.dni LIKE :query
OR u.email LIKE :query)
LIMIT 10";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([
':institution_id' => $_SESSION['institution_id'],
':query' => '%' . $query . '%'
]);

header('Content-Type: application/json');
echo json_encode($stmt->fetchAll());
exit;
}

public function advancedAttendanceReport() {
if (!Security::hasRole(['autoridad', 'inspector'])) {
die('Acceso denegado');
}

```

```php
$filters = [
'course_id' => $_POST['course_id'] ?? null,
'student_id' => $_POST['student_id'] ?? null,
'subject_id' => $_POST['subject_id'] ?? null,
'status' => $_POST['status'] ?? null,
'start_date' => $_POST['start_date'] ?? null,
'end_date' => $_POST['end_date'] ?? null
];

$results = $this->attendanceModel->getFiltered($filters);

include BASE_PATH . '/views/reports/advanced.php';
}
}
```

4.16 — StatsController.php

controllers/StatsController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/Course.php';
require_once BASE_PATH . '/models/User.php';

class StatsController {
private $attendanceModel;
private $courseModel;
private $userModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$db = new Database();
$this->attendanceModel = new Attendance($db);
$this->courseModel = new Course($db);
$this->userModel = new User($db);
}

public function index() {
$startDate = $_GET['start_date'] ?? date('Y-m-01');
$endDate = $_GET['end_date'] ?? date('Y-m-d');

$stats = [
'overall' => $this->getOverallStats($startDate, $endDate),
'by_course' => $this->getStatsByCourse($startDate, $endDate),
'by_student' => $this->getTopAbsentStudents($startDate, $endDate),
'by_day' => $this->getStatsByDay($startDate, $endDate)
];

include BASE_PATH . '/views/stats/index.php';
}

private function getOverallStats($startDate, $endDate) {
$sql = "SELECT
COUNT(*) as total_records,
COUNT(DISTINCT student_id) as total_students,
SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN status = 'ausente' THEN 1 ELSE 0 END) as ausente,
SUM(CASE WHEN status = 'tardanza' THEN 1 ELSE 0 END) as tardanza,
SUM(CASE WHEN status = 'justificado' THEN 1 ELSE 0 END) as justificado
FROM attendances
WHERE date BETWEEN :start_date AND :end_date";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':start_date' => $startDate, ':end_date' => $endDate]);
return $stmt->fetch();
}

private function getStatsByCourse($startDate, $endDate) {
$sql = "SELECT
c.name as course_name,
COUNT(*) as total,
SUM(CASE WHEN a.status = 'presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN a.status = 'ausente' THEN 1 ELSE 0 END) as ausente,
```

```
ROUND(SUM(CASE WHEN a.status = 'presente' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) as percentage
FROM attendances a
INNER JOIN courses c ON a.course_id = c.id
WHERE a.date BETWEEN :start_date AND :end_date
GROUP BY c.id, c.name
ORDER BY percentage DESC";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':start_date' => $startDate, ':end_date' => $endDate]);
return $stmt->fetchAll();
}

private function getTopAbsentStudents($startDate, $endDate, $limit = 10) {
$sql = "SELECT
CONCAT(u.last_name, ' ', u.first_name) as student_name,
c.name as course_name,
COUNT(*) as total_absences
FROM attendances a
INNER JOIN users u ON a.student_id = u.id
INNER JOIN courses c ON a.course_id = c.id
WHERE a.status = 'ausente'
AND a.date BETWEEN :start_date AND :end_date
GROUP BY a.student_id, u.last_name, u.first_name, c.name
ORDER BY total_absences DESC
LIMIT :limit";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->bindValue(':start_date', $startDate);
$stmt->bindValue(':end_date', $endDate);
$stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
$stmt->execute();
return $stmt->fetchAll();
}

private function getStatsByDay($startDate, $endDate) {
$sql = "SELECT
date,
COUNT(*) as total,
SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
ROUND(SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) as percentage
FROM attendances
WHERE date BETWEEN :start_date AND :end_date
GROUP BY date
ORDER BY date ASC";

$db = new Database();
$stmt = $db->connect()->prepare($sql);
$stmt->execute([':start_date' => $startDate, ':end_date' => $endDate]);
return $stmt->fetchAll();
}
}
```

4.17 — TutorController.php

controllers/TutorController.php PHP

```php
<?php
// ARCHIVO COMPLETO - Reemplaza TODO el archivo
// controllers/TutorController.php

require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/Attendance.php';
require_once BASE_PATH . '/models/Course.php';

class TutorController {
private $attendanceModel;
private $courseModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('docente')) {
die('Acceso denegado');
}
$db = new Database();
$this->attendanceModel = new Attendance($db);
$this->courseModel = new Course($db);
}

public function courseAttendance() {
$teacherId = $_SESSION['user_id'];
$courseId = $this->attendanceModel->getTutorCourseId($teacherId);

if (!$courseId) {
include BASE_PATH . '/views/tutor/no_tutor.php';
return;
}

$course = $this->courseModel->findById($courseId);
$subjects = $this->attendanceModel->getSubjectsByCourse($courseId);
$students = $this->attendanceModel->getStudentsByCourse($courseId);
$topAbsences = $this->attendanceModel->getTutorTopAbsences($courseId, 5);

// Leer filtros del GET
$filters = [];
if (!empty($_GET['subject_id'])) $filters['subject_id'] = (int)$_GET['subject_id'];
if (!empty($_GET['student_id'])) $filters['student_id'] = (int)$_GET['student_id'];
if (!empty($_GET['status'])) $filters['status'] = $_GET['status'];
if (!empty($_GET['start_date'])) $filters['start_date'] = $_GET['start_date'];
if (!empty($_GET['end_date'])) $filters['end_date'] = $_GET['end_date'];

$attendances = $this->attendanceModel->getTutorCourseAttendance($courseId, $filters);
$stats = $this->attendanceModel->getTutorCourseStats($courseId, $filters);

include BASE_PATH . '/views/tutor/course_attendance.php';
}

// Endpoint AJAX
public function ajax() {
header('Content-Type: application/json');

$teacherId = $_SESSION['user_id'];
$courseId = $this->attendanceModel->getTutorCourseId($teacherId);

if (!$courseId) {
echo json_encode(['error' => 'No es tutor']);
exit;
```

```
}

$filters = [];
if (!empty($_GET['subject_id'])) $filters['subject_id'] = (int)$_GET['subject_id'];
if (!empty($_GET['student_id'])) $filters['student_id'] = (int)$_GET['student_id'];
if (!empty($_GET['status'])) $filters['status'] = $_GET['status'];
if (!empty($_GET['start_date'])) $filters['start_date'] = $_GET['start_date'];
if (!empty($_GET['end_date'])) $filters['end_date'] = $_GET['end_date'];

$attendances = $this->attendanceModel->getTutorCourseAttendance($courseId, $filters);
$stats = $this->attendanceModel->getTutorCourseStats($courseId, $filters);

echo json_encode([
'stats' => $stats,
'attendances' => $attendances,
'total' => count($attendances),
]);
exit;
}

// ■■ Dashboard del tutor ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
public function dashboard() {
$teacherId = $_SESSION['user_id'];
$courseId = $this->attendanceModel->getTutorCourseId($teacherId);

if (!$courseId) {
include BASE_PATH . '/views/tutor/no_tutor.php';
return;
}

$course = $this->courseModel->findById($courseId);
$students = $this->attendanceModel->getStudentsByCourse($courseId);
$topAbsences = $this->attendanceModel->getTutorTopAbsences($courseId, 10);
$stats = $this->attendanceModel->getTutorCourseStats($courseId);

// Asistencia de hoy
$db = new Database();
$pdo = $db->connect();
$stmt = $pdo->prepare(
"SELECT COUNT(*) as total,
SUM(CASE WHEN status='presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN status='ausente' THEN 1 ELSE 0 END) as ausente,
SUM(CASE WHEN status='tardanza' THEN 1 ELSE 0 END) as tardanza,
SUM(CASE WHEN status='justificado' THEN 1 ELSE 0 END) as justificado
FROM attendances
WHERE course_id = :course_id AND date = CURDATE()"
);
$stmt->execute([':course_id' => $courseId]);
$todayStats = $stmt->fetch();

// Tendencia últimos 7 días
$stmt2 = $pdo->prepare(
"SELECT date,
COUNT(*) as total,
SUM(CASE WHEN status='presente' THEN 1 ELSE 0 END) as presente
FROM attendances
WHERE course_id = :course_id
AND date >= DATE_SUB(CURDATE(), INTERVAL 6 DAY)
GROUP BY date
ORDER BY date ASC"
```

```
);
$stmt2->execute([':course_id' => $courseId]);
$trend = $stmt2->fetchAll();

include BASE_PATH . '/views/tutor/dashboard.php';
}

// ■■ Buscar estudiantes del curso ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
public function searchStudents() {
$teacherId = $_SESSION['user_id'];
$courseId = $this->attendanceModel->getTutorCourseId($teacherId);

if (!$courseId) {
include BASE_PATH . '/views/tutor/no_tutor.php';
return;
}

$course = $this->courseModel->findById($courseId);
$query = trim($_GET['q'] ?? '');
$students = [];

$db = new Database();
$pdo = $db->connect();

$sql = "SELECT u.id, u.first_name, u.last_name, u.email, u.dni, u.phone,
COUNT(a.id) as total_clases,
SUM(CASE WHEN a.status='presente' THEN 1 ELSE 0 END) as presentes,
SUM(CASE WHEN a.status='ausente' THEN 1 ELSE 0 END) as ausentes,
SUM(CASE WHEN a.status='tardanza' THEN 1 ELSE 0 END) as tardanzas,
SUM(CASE WHEN a.status='justificado' THEN 1 ELSE 0 END) as justificados
FROM users u
INNER JOIN course_students cs ON u.id = cs.student_id
LEFT JOIN attendances a ON a.student_id = u.id AND a.course_id = :course_id
WHERE cs.course_id = :course_id2";

$params = [':course_id' => $courseId, ':course_id2' => $courseId];

if ($query !== '') {
$sql .= " AND (u.first_name LIKE :q OR u.last_name LIKE :q2 OR u.dni LIKE :q3
OR CONCAT(u.last_name,' ',u.first_name) LIKE :q4)";
$like = '%' . $query . '%';
$params[':q'] = $like;
$params[':q2'] = $like;
$params[':q3'] = $like;
$params[':q4'] = $like;
}

$sql .= " GROUP BY u.id, u.first_name, u.last_name, u.email, u.dni, u.phone
ORDER BY u.last_name, u.first_name";

$stmt = $pdo->prepare($sql);
$stmt->execute($params);
$students = $stmt->fetchAll();

include BASE_PATH . '/views/tutor/search_students.php';
}
}
```

4.18 — UserController.php

controllers/UserController.php PHP

```php
<?php
require_once BASE_PATH . '/config/config.php';
require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/Role.php';

class UserController {
private $userModel;
private $roleModel;

public function __construct() {
Security::requireLogin();
if (!Security::hasRole('autoridad')) {
die('Acceso denegado');
}

$db = new Database();
$this->userModel = new User($db);
$this->roleModel = new Role($db);
}

public function index() {
$users = $this->userModel->getAll();
$roles = $this->roleModel->getAll();
include BASE_PATH . '/views/users/index.php';
}

public function assignRole() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$userId = (int)$_POST['user_id'];
$roleId = (int)$_POST['role_id'];

$this->userModel->assignRole($userId, $roleId);
$filter = isset($_GET['filter_role']) ? '&filter_role=' . $_GET['filter_role'] : '';
header('Location: ?action=users&success=1' . $filter);
exit;
}
}

public function removeRole() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$userId = (int)$_POST['user_id'];
$roleId = (int)$_POST['role_id'];

// Verificar si el rol es "docente"
$db = new Database();
$stmt = $db->connect()->prepare("SELECT name FROM roles WHERE id = :role_id");
$stmt->execute([':role_id' => $roleId]);
$role = $stmt->fetch();

if ($role && $role['name'] === 'docente') {
// Eliminar asignaciones de materias
$stmt = $db->connect()->prepare("DELETE FROM teacher_assignments WHERE teacher_id = :user_id");
$stmt->execute([':user_id' => $userId]);
}

$this->userModel->removeRole($userId, $roleId);
$filter = isset($_GET['filter_role']) ? '&filter_role=' . $_GET['filter_role'] : '';
header('Location: ?action=users&removed=1' . $filter);
exit;
}
```

```
}

public function create() {
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
$roles = $this->roleModel->getAll();
include BASE_PATH . '/views/users/create.php';
return;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$errors = [];

// Validaciones
if (empty($_POST['username'])) {
$errors[] = "El nombre de usuario es obligatorio";
}
if (empty($_POST['email']) || !filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
$errors[] = "Email inválido";
}
if (empty($_POST['password']) || strlen($_POST['password']) < 6) {
$errors[] = "La contraseña debe tener al menos 6 caracteres";
}
if ($_POST['password'] !== $_POST['confirm_password']) {
$errors[] = "Las contraseñas no coinciden";
}
if (empty($_POST['first_name'])) {
$errors[] = "El nombre es obligatorio";
}
if (empty($_POST['last_name'])) {
$errors[] = "El apellido es obligatorio";
}

// Verificar username único
if ($this->userModel->findByUsername($_POST['username'])) {
$errors[] = "El nombre de usuario ya está registrado";
}

// Verificar email único
if ($this->userModel->findByEmail($_POST['email'])) {
$errors[] = "El email ya está registrado";
}

if (!empty($errors)) {
$roles = $this->roleModel->getAll();
include BASE_PATH . '/views/users/create.php';
return;
}

// Crear usuario
$userData = [
'institution_id' => $_SESSION['institution_id'],
'username' => Security::sanitize($_POST['username']),
'email' => Security::sanitize($_POST['email']),
'password' => $_POST['password'],
'first_name' => Security::sanitize($_POST['first_name']),
'last_name' => Security::sanitize($_POST['last_name']),
'dni' => !empty($_POST['dni']) ? Security::sanitize($_POST['dni']) : null,
'phone' => !empty($_POST['phone']) ? Security::sanitize($_POST['phone']) : null
];

```

```
if ($this->userModel->create($userData)) {
// Obtener ID del usuario recién creado
$newUserId = $this->userModel->db->lastInsertId();

// Asignar roles si se seleccionaron
if (!empty($_POST['roles'])) {
foreach ($_POST['roles'] as $roleId) {
$this->userModel->assignRole($newUserId, (int)$roleId);
}
}

header('Location: ?action=users&created=1');
exit;
} else {
$errors[] = "Error al crear el usuario";
$roles = $this->roleModel->getAll();
include BASE_PATH . '/views/users/create.php';
}
}
}

public function edit() {
$userId = (int)$_GET['id'];
$user = $this->userModel->findById($userId);

if (!$user) {
header('Location: ?action=users&error=not_found');
exit;
}

// Verificar que pertenece a la misma institución
if ($user['institution_id'] != $_SESSION['institution_id']) {
die('Acceso denegado');
}

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
$roles = $this->roleModel->getAll();
$userRoles = $this->userModel->getUserRoleIds($userId);
include BASE_PATH . '/views/users/edit.php';
return;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$errors = [];

// Validaciones
if (empty($_POST['username'])) {
$errors[] = "El nombre de usuario es obligatorio";
}
if (empty($_POST['email']) || !filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
$errors[] = "Email inválido";
}
if (empty($_POST['first_name'])) {
$errors[] = "El nombre es obligatorio";
}
if (empty($_POST['last_name'])) {
$errors[] = "El apellido es obligatorio";
}

// Verificar username único (excepto el actual)
```

```
$existingUser = $this->userModel->findByUsername($_POST['username']);
if ($existingUser && $existingUser['id'] != $userId) {
$errors[] = "El nombre de usuario ya está registrado";
}

// Verificar email único (excepto el actual)
$existingEmail = $this->userModel->findByEmail($_POST['email']);
if ($existingEmail && $existingEmail['id'] != $userId) {
$errors[] = "El email ya está registrado";
}

// Si hay nueva contraseña, validarla
if (!empty($_POST['password'])) {
if (strlen($_POST['password']) < 6) {
$errors[] = "La contraseña debe tener al menos 6 caracteres";
}
if ($_POST['password'] !== $_POST['confirm_password']) {
$errors[] = "Las contraseñas no coinciden";
}
}

if (!empty($errors)) {
$roles = $this->roleModel->getAll();
$userRoles = $this->userModel->getUserRoleIds($userId);
include BASE_PATH . '/views/users/edit.php';
return;
}

// Actualizar usuario
$updateData = [
'id' => $userId,
'username' => Security::sanitize($_POST['username']),
'email' => Security::sanitize($_POST['email']),
'first_name' => Security::sanitize($_POST['first_name']),
'last_name' => Security::sanitize($_POST['last_name']),
'dni' => !empty($_POST['dni']) ? Security::sanitize($_POST['dni']) : null,
'phone' => !empty($_POST['phone']) ? Security::sanitize($_POST['phone']) : null
];

// Solo actualizar contraseña si se proporcionó una nueva
if (!empty($_POST['password'])) {
$updateData['password'] = $_POST['password'];
}

if ($this->userModel->update($updateData)) {
header('Location: ?action=users&updated=1');
exit;
} else {
$errors[] = "Error al actualizar el usuario";
$roles = $this->roleModel->getAll();
$userRoles = $this->userModel->getUserRoleIds($userId);
include BASE_PATH . '/views/users/edit.php';
}
}
}

public function delete() {
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$userId = (int)$_POST['user_id'];
$user = $this->userModel->findById($userId);
```

```
241
242 if (!$user) {
243 header('Location: ?action=users&error=not_found');
244 exit;
245 }
246
247 // Verificar que pertenece a la misma institución
248 if ($user['institution_id'] != $_SESSION['institution_id']) {
249 die('Acceso denegado');
250 }
251
252 // No permitir eliminar al propio usuario
253 if ($userId == $_SESSION['user_id']) {
254 header('Location: ?action=users&error=self_delete');
255 exit;
256 }
257
258 // Verificar si tiene asistencias registradas
259 $db = new Database();
260 $stmt = $db->connect()->prepare("SELECT COUNT(*) as count FROM attendances WHERE student_id = :user_id");
261 $stmt->execute([':user_id' => $userId]);
262 $result = $stmt->fetch();
263
264 if ($result['count'] > 0) {
265 // Soft delete - desactivar en lugar de eliminar
266 if ($this->userModel->deactivate($userId)) {
267 header('Location: ?action=users&deactivated=1');
268 exit;
269 }
270 } else {
271 // Hard delete - eliminar completamente
272 if ($this->userModel->delete($userId)) {
273 header('Location: ?action=users&deleted=1');
274 exit;
275 }
276 }
277
278 header('Location: ?action=users&error=delete_failed');
279 exit;
280 }
281 }
282 }
```

5. MODULO MODELS (14 archivos)

5.1 — Attendance.php

models/Attendance.php PHP

```php
<?php
// ARCHIVO COMPLETO - Reemplaza TODO el archivo
// models/Attendance.php

class Attendance {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function create($data) {
$checkSql = "SELECT id FROM attendances
WHERE student_id = :student_id
AND course_id = :course_id
AND subject_id = :subject_id
AND date = :date
AND hour_period = :hour_period";
$checkStmt = $this->db->prepare($checkSql);
$checkStmt->execute([
':student_id' => $data[':student_id'],
':course_id' => $data[':course_id'],
':subject_id' => $data[':subject_id'],
':date' => $data[':date'],
':hour_period' => $data[':hour_period']
]);
$existing = $checkStmt->fetch();

if ($existing) {
$updateSql = "UPDATE attendances
SET status = :status, observation = :observation, updated_at = NOW()
WHERE id = :id";
$updateStmt = $this->db->prepare($updateSql);
return $updateStmt->execute([
':status' => $data[':status'],
':observation' => $data[':observation'],
':id' => $existing['id']
]);
}

$sql = "INSERT INTO attendances
(student_id, course_id, subject_id, teacher_id, school_year_id, shift_id, date, hour_period, status,
observation)
VALUES (:student_id, :course_id, :subject_id, :teacher_id, :school_year_id, :shift_id, :date, :hour_period,
:status, :observation)";
$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function canEdit($attendanceId) {
$sql = "SELECT created_at FROM attendances WHERE id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $attendanceId]);
$record = $stmt->fetch();
if (!$record) return false;
return (time() - strtotime($record['created_at'])) <= (EDIT_ATTENDANCE_HOURS * 3600);
}

public function update($id, $data) {
$sql = "UPDATE attendances
SET status = :status, observation = :observation, updated_at = NOW()
WHERE id = :id";
```

```
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $id, ':status' => $data['status'], ':observation' => $data['observation']]);
}

public function getByCourse($courseId, $date) {
$sql = "SELECT a.*, u.first_name, u.last_name, s.name as subject_name,
CONCAT(t.first_name, ' ', t.last_name) as teacher_name
FROM attendances a
INNER JOIN users u ON a.student_id = u.id
INNER JOIN subjects s ON a.subject_id = s.id
INNER JOIN users t ON a.teacher_id = t.id
WHERE a.course_id = :course_id AND a.date = :date
ORDER BY u.last_name, u.first_name";
$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId, ':date' => $date]);
return $stmt->fetchAll();
}

public function getByStudent($studentId) {
$sql = "SELECT a.*, s.name as subject_name, c.name as course_name, sh.name as shift_name
FROM attendances a
INNER JOIN subjects s ON a.subject_id = s.id
INNER JOIN courses c ON a.course_id = c.id
INNER JOIN shifts sh ON a.shift_id = sh.id
WHERE a.student_id = :student_id
ORDER BY a.date DESC, a.hour_period
LIMIT 100";
$stmt = $this->db->prepare($sql);
$stmt->execute([':student_id' => $studentId]);
return $stmt->fetchAll();
}

public function getReportData($courseId, $startDate, $endDate) {
$sql = "SELECT a.*,
CONCAT(u.last_name, ' ', u.first_name) as student_name,
s.name as subject_name
FROM attendances a
INNER JOIN users u ON a.student_id = u.id
INNER JOIN subjects s ON a.subject_id = s.id
WHERE a.course_id = :course_id
AND a.date BETWEEN :start_date AND :end_date
ORDER BY a.date, u.last_name, u.first_name";
$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId, ':start_date' => $startDate, ':end_date' => $endDate]);
return $stmt->fetchAll();
}

public function getStatsByCourse($courseId, $startDate, $endDate) {
$sql = "SELECT
COUNT(*) as total,
SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN status = 'ausente' THEN 1 ELSE 0 END) as ausente,
SUM(CASE WHEN status = 'tardanza' THEN 1 ELSE 0 END) as tardanza,
SUM(CASE WHEN status = 'justificado' THEN 1 ELSE 0 END) as justificado
FROM attendances
WHERE course_id = :course_id AND date BETWEEN :start_date AND :end_date";
$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId, ':start_date' => $startDate, ':end_date' => $endDate]);
return $stmt->fetch();
}
```

```
121
122 public function getFiltered($filters) {
123 $sql = "SELECT a.*,
124 CONCAT(u.last_name, ' ', u.first_name) as student_name,
125 c.name as course_name, s.name as subject_name,
126 CONCAT(t.last_name, ' ', t.first_name) as teacher_name
127 FROM attendances a
128 INNER JOIN users u ON a.student_id = u.id
129 INNER JOIN courses c ON a.course_id = c.id
130 INNER JOIN subjects s ON a.subject_id = s.id
131 INNER JOIN users t ON a.teacher_id = t.id
132 WHERE 1=1";
133 $params = [];
134 if (!empty($filters['course_id'])) { $sql .= " AND a.course_id = :course_id"; $params[':course_id'] =
$filters['course_id']; }
135 if (!empty($filters['student_id'])) { $sql .= " AND a.student_id = :student_id"; $params[':student_id'] =
$filters['student_id']; }
136 if (!empty($filters['subject_id'])) { $sql .= " AND a.subject_id = :subject_id"; $params[':subject_id'] =
$filters['subject_id']; }
137 if (!empty($filters['status'])) { $sql .= " AND a.status = :status"; $params[':status'] =
$filters['status']; }
138 if (!empty($filters['start_date'])) { $sql .= " AND a.date >= :start_date"; $params[':start_date'] =
$filters['start_date']; }
139 if (!empty($filters['end_date'])) { $sql .= " AND a.date <= :end_date"; $params[':end_date'] =
$filters['end_date']; }
140 $sql .= " ORDER BY a.date DESC, u.last_name, u.first_name";
141 $stmt = $this->db->prepare($sql);
142 $stmt->execute($params);
143 return $stmt->fetchAll();
144 }
145
146 public function getDayStats($courseId, $date) {
147 $sql = "SELECT COUNT(*) as total,
148 SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
149 SUM(CASE WHEN status = 'ausente' THEN 1 ELSE 0 END) as ausente,
150 SUM(CASE WHEN status = 'tardanza' THEN 1 ELSE 0 END) as tardanza
151 FROM attendances WHERE course_id = :course_id AND date = :date";
152 $stmt = $this->db->prepare($sql);
153 $stmt->execute([':course_id' => $courseId, ':date' => $date]);
154 return $stmt->fetch();
155 }
156
157 // ■■ MÉTODOS PARA DOCENTE TUTOR ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
158
159 // Obtener el course_id donde el docente es tutor en el año activo
160 public function getTutorCourseId($teacherId) {
161 $sql = "SELECT ta.course_id
162 FROM teacher_assignments ta
163 INNER JOIN school_years sy ON ta.school_year_id = sy.id
164 WHERE ta.teacher_id = :teacher_id AND ta.is_tutor = 1 AND sy.is_active = 1
165 LIMIT 1";
166 $stmt = $this->db->prepare($sql);
167 $stmt->execute([':teacher_id' => $teacherId]);
168 $row = $stmt->fetch();
169 return $row ? (int)$row['course_id'] : null;
170 }
171
172 // Asignaturas del curso (para filtro)
173 public function getSubjectsByCourse($courseId) {
174 $sql = "SELECT DISTINCT s.id, s.name
175 FROM subjects s
176 INNER JOIN teacher_assignments ta ON s.id = ta.subject_id
177 WHERE ta.course_id = :course_id
178 ORDER BY s.name";
179 $stmt = $this->db->prepare($sql);
180 $stmt->execute([':course_id' => $courseId]);
```

```
return $stmt->fetchAll();
}

// Estudiantes del curso (para filtro)
public function getStudentsByCourse($courseId) {
$sql = "SELECT u.id, CONCAT(u.last_name, ' ', u.first_name) as full_name
FROM users u
INNER JOIN course_students cs ON u.id = cs.student_id
WHERE cs.course_id = :course_id
ORDER BY u.last_name, u.first_name";
$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId]);
return $stmt->fetchAll();
}

// Asistencias del curso con filtros (vista del tutor)
public function getTutorCourseAttendance($courseId, $filters = []) {
$sql = "SELECT
a.id, a.date, a.hour_period, a.status, a.observation,
CONCAT(u.last_name, ' ', u.first_name) as student_name,
u.id as student_id,
s.name as subject_name,
s.id as subject_id,
CONCAT(t.last_name, ' ', t.first_name) as teacher_name
FROM attendances a
INNER JOIN users u ON a.student_id = u.id
INNER JOIN subjects s ON a.subject_id = s.id
INNER JOIN users t ON a.teacher_id = t.id
WHERE a.course_id = :course_id";
$params = [':course_id' => $courseId];

if (!empty($filters['subject_id'])) { $sql .= " AND a.subject_id = :subject_id"; $params[':subject_id'] = $filters['subject_id']; }
if (!empty($filters['student_id'])) { $sql .= " AND a.student_id = :student_id"; $params[':student_id'] = $filters['student_id']; }
if (!empty($filters['status'])) { $sql .= " AND a.status = :status"; $params[':status'] = $filters['status']; }
if (!empty($filters['start_date'])) { $sql .= " AND a.date >= :start_date"; $params[':start_date'] = $filters['start_date']; }
if (!empty($filters['end_date'])) { $sql .= " AND a.date <= :end_date"; $params[':end_date'] = $filters['end_date']; }

$sql .= " ORDER BY a.date DESC, u.last_name, u.first_name, s.name";
$stmt = $this->db->prepare($sql);
$stmt->execute($params);
return $stmt->fetchAll();
}

// Estadísticas globales del curso (vista del tutor)
public function getTutorCourseStats($courseId, $filters = []) {
$sql = "SELECT
COUNT(*) as total,
SUM(CASE WHEN status = 'presente' THEN 1 ELSE 0 END) as presente,
SUM(CASE WHEN status = 'ausente' THEN 1 ELSE 0 END) as ausente,
SUM(CASE WHEN status = 'tardanza' THEN 1 ELSE 0 END) as tardanza,
SUM(CASE WHEN status = 'justificado' THEN 1 ELSE 0 END) as justificado
FROM attendances a
WHERE a.course_id = :course_id";
$params = [':course_id' => $courseId];

if (!empty($filters['subject_id'])) { $sql .= " AND a.subject_id = :subject_id"; $params[':subject_id'] = $filters['subject_id']; }
if (!empty($filters['student_id'])) { $sql .= " AND a.student_id = :student_id"; $params[':student_id'] = $filters['student_id']; }
if (!empty($filters['start_date'])) { $sql .= " AND a.date >= :start_date"; $params[':start_date'] = $filters['start_date']; }
if (!empty($filters['end_date'])) { $sql .= " AND a.date <= :end_date"; $params[':end_date'] = $filters['end_date']; }

```

```php
$stmt = $this->db->prepare($sql);
$stmt->execute($params);
return $stmt->fetch();
}

// Top estudiantes con más ausencias (vista del tutor)
public function getTutorTopAbsences($courseId, $limit = 10) {
$sql = "SELECT CONCAT(u.last_name, ' ', u.first_name) as student_name,
COUNT(*) as total_ausencias
FROM attendances a
INNER JOIN users u ON a.student_id = u.id
WHERE a.course_id = :course_id AND a.status = 'ausente'
GROUP BY a.student_id, u.last_name, u.first_name
ORDER BY total_ausencias DESC
LIMIT :lim";
$stmt = $this->db->prepare($sql);
$stmt->bindValue(':course_id', $courseId, PDO::PARAM_INT);
$stmt->bindValue(':lim', $limit, PDO::PARAM_INT);
$stmt->execute();
return $stmt->fetchAll();
}

// Ausencias sin justificación pendiente o aprobada (para el formulario de justificación)
public function getUnjustifiedAbsences($studentId) {
$sql = "SELECT a.id, a.date, a.hour_period, a.status,
s.name as subject_name,
c.name as course_name
FROM attendances a
INNER JOIN subjects s ON a.subject_id = s.id
INNER JOIN courses c ON a.course_id = c.id
WHERE a.student_id = :student_id
AND a.status = 'ausente'
AND a.id NOT IN (
SELECT attendance_id FROM justifications
WHERE attendance_id IS NOT NULL
AND status IN ('pendiente','aprobado')
)
ORDER BY a.date DESC, a.hour_period ASC";
$stmt = $this->db->prepare($sql);
$stmt->execute([':student_id' => $studentId]);
return $stmt->fetchAll();
}
}
```

5.2 — ClassSchedule.php

models/ClassSchedule.php PHP

```php
<?php
class ClassSchedule {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function create($data) {
// Verificar si ya existe una clase en ese horario
$checkSql = "SELECT s.name as subject_name,
CONCAT(u.last_name, ' ', u.first_name) as teacher_name
FROM class_schedule cs
INNER JOIN subjects s ON cs.subject_id = s.id
INNER JOIN users u ON cs.teacher_id = u.id
WHERE cs.course_id = :course_id
AND cs.day_of_week = :day_of_week
AND cs.period_number = :period_number
AND cs.school_year_id = :school_year_id";

$checkStmt = $this->db->prepare($checkSql);
$checkStmt->execute([
':course_id' => $data[':course_id'],
':day_of_week' => $data[':day_of_week'],
':period_number' => $data[':period_number'],
':school_year_id' => $data[':school_year_id']
]);

$existing = $checkStmt->fetch();

if ($existing) {
return [
'success' => false,
'message' => 'Ya existe ' . $existing['subject_name'] . ' con ' . $existing['teacher_name'] . ' en este
horario'
];
}

$sql = "INSERT INTO class_schedule
(course_id, subject_id, teacher_id, school_year_id, day_of_week, period_number)
VALUES (:course_id, :subject_id, :teacher_id, :school_year_id, :day_of_week, :period_number)";

$stmt = $this->db->prepare($sql);
$result = $stmt->execute($data);

return [
'success' => $result,
'message' => 'Clase agregada correctamente'
];
}

public function getByCourse($courseId, $schoolYearId) {
$sql = "SELECT cs.*,
s.name as subject_name,
CONCAT(u.last_name, ' ', u.first_name) as teacher_name
FROM class_schedule cs
INNER JOIN subjects s ON cs.subject_id = s.id
INNER JOIN users u ON cs.teacher_id = u.id
WHERE cs.course_id = :course_id
AND cs.school_year_id = :school_year_id
ORDER BY cs.day_of_week, cs.period_number";
```

```
61
62 $stmt = $this->db->prepare($sql);
63 $stmt->execute([
64 ':course_id' => $courseId,
65 ':school_year_id' => $schoolYearId
66 ]);
67 return $stmt->fetchAll();
68 }
69
70 public function getTeacherScheduleToday($teacherId, $schoolYearId) {
71 $dayName = $this->getCurrentDayName();
72
73 $sql = "SELECT cs.*,
74 c.name as course_name,
75 c.grade_level,
76 s.name as subject_name
77 FROM class_schedule cs
78 INNER JOIN courses c ON cs.course_id = c.id
79 INNER JOIN subjects s ON cs.subject_id = s.id
80 WHERE cs.teacher_id = :teacher_id
81 AND cs.school_year_id = :school_year_id
82 AND cs.day_of_week = :day_of_week
83 ORDER BY cs.period_number";
84
85 $stmt = $this->db->prepare($sql);
86 $stmt->execute([
87 ':teacher_id' => $teacherId,
88 ':school_year_id' => $schoolYearId,
89 ':day_of_week' => $dayName
90 ]);
91 return $stmt->fetchAll();
92 }
93
94 public function delete($id) {
95 $sql = "DELETE FROM class_schedule WHERE id = :id";
96 $stmt = $this->db->prepare($sql);
97 return $stmt->execute([':id' => $id]);
98 }
99
100 private function getCurrentDayName() {
101 // Forzar zona horaria Ecuador
102 $tz = new DateTimeZone('America/Guayaquil');
103 $now = new DateTime('now', $tz);
104 $dayNumber = (int)$now->format('N'); // 1=Lunes, 7=Domingo
105
106 $days = [
107 1 => 'lunes',
108 2 => 'martes',
109 3 => 'miercoles',
110 4 => 'jueves',
111 5 => 'viernes',
112 6 => 'sabado'
113 ];
114
115 return $days[$dayNumber] ?? 'lunes';
116 }
117 }
```

5.3 — Course.php

models/Course.php PHP

```php
<?php
class Course {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function getAll() {
$sql = "SELECT c.*, s.name as shift_name, sy.name as year_name
FROM courses c
INNER JOIN shifts s ON c.shift_id = s.id
INNER JOIN school_years sy ON c.school_year_id = sy.id
WHERE c.institution_id = :institution_id
ORDER BY c.name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $_SESSION['institution_id']]);
return $stmt->fetchAll();
}

public function create($data) {
$sql = "INSERT INTO courses (institution_id, school_year_id, name, grade_level, parallel, shift_id)
VALUES (:institution_id, :school_year_id, :name, :grade_level, :parallel, :shift_id)";

$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function getStudents($courseId) {
$sql = "SELECT u.id, u.first_name, u.last_name, u.dni
FROM users u
INNER JOIN course_students cs ON u.id = cs.student_id
WHERE cs.course_id = :course_id
ORDER BY u.last_name, u.first_name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId]);
return $stmt->fetchAll();
}

public function enrollStudent($courseId, $studentId, $schoolYearId) {
// Primero verificar si ya está matriculado en otro curso del mismo año
$checkSql = "SELECT COUNT(*) as count FROM course_students
WHERE student_id = :student_id AND school_year_id = :school_year_id";
$checkStmt = $this->db->prepare($checkSql);
$checkStmt->execute([
':student_id' => $studentId,
':school_year_id' => $schoolYearId
]);
$result = $checkStmt->fetch();

if ($result['count'] > 0) {
return false; // Ya está matriculado
}

$sql = "INSERT INTO course_students (student_id, course_id, school_year_id, enrollment_date)
VALUES (:student_id, :course_id, :school_year_id, CURDATE())";

$stmt = $this->db->prepare($sql);
```

```
return $stmt->execute([
':student_id' => $studentId,
':course_id' => $courseId,
':school_year_id' => $schoolYearId
]);
}

public function getEnrolledStudents($courseId) {
$sql = "SELECT u.id, u.first_name, u.last_name, u.dni, cs.enrollment_date
FROM users u
INNER JOIN course_students cs ON u.id = cs.student_id
WHERE cs.course_id = :course_id
ORDER BY u.last_name, u.first_name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId]);
return $stmt->fetchAll();
}

public function findById($id) {
$sql = "SELECT c.*, s.name as shift_name
FROM courses c
INNER JOIN shifts s ON c.shift_id = s.id
WHERE c.id = :id";

$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);
return $stmt->fetch();
}

public function update($data) {
$sql = "UPDATE courses SET
name = :name,
grade_level = :grade_level,
parallel = :parallel,
shift_id = :shift_id,
updated_at = NOW()
WHERE id = :id";

$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function delete($id) {
// Eliminar relaciones primero
$sql = "DELETE FROM course_students WHERE course_id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);

// Eliminar horarios
$sql = "DELETE FROM class_schedule WHERE course_id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);

// Eliminar curso
$sql = "DELETE FROM courses WHERE id = :id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $id]);
}

```

```php
public function unenrollStudent($studentId, $schoolYearId) {
$sql = "DELETE FROM course_students
WHERE student_id = :student_id AND school_year_id = :school_year_id";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':student_id' => $studentId,
':school_year_id' => $schoolYearId
]);
}
}
```

5.4 — Institution.php

models/Institution.php PHP

```php
<?php
class Institution {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function getById($id) {
$sql = "SELECT * FROM institutions WHERE id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);
return $stmt->fetch();
}

public function getAll() {
$sql = "SELECT * FROM institutions ORDER BY name";
$stmt = $this->db->query($sql);
return $stmt->fetchAll();
}

public function update($id, $data) {
$sql = "UPDATE institutions SET
name = :name,
address = :address,
province = :province,
city = :city,
phone = :phone,
email = :email,
director_name = :director_name,
amie_code = :amie_code,
website = :website,
logo_path = :logo_path,
updated_at = NOW()
WHERE id = :id";

// Obtener logo actual si no se proporciona uno nuevo
$currentInstitution = $this->getById($id);
$logoPath = $data['logo_path'] ?? $currentInstitution['logo_path'];

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':id' => $id,
':name' => $data['name'],
':address' => $data['address'],
':province' => $data['province'],
':city' => $data['city'],
':phone' => $data['phone'],
':email' => $data['email'],
':director_name' => $data['director_name'],
':amie_code' => $data['amie_code'],
':website' => $data['website'],
':logo_path' => $logoPath
]);
}

public function getShifts($institutionId) {
$sql = "SELECT s.id, s.name,
GROUP_CONCAT(s.name ORDER BY s.name) as shift_names
FROM shifts s
```

```
INNER JOIN institution_shifts ins ON s.id = ins.shift_id
WHERE ins.institution_id = :institution_id
GROUP BY ins.institution_id";

$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $institutionId]);
return $stmt->fetch();
}
}
```

5.5 — InstitutionShift.php

models/InstitutionShift.php PHP

```
<?php
class InstitutionShift {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function assign($data) {
$sql = "INSERT INTO institution_shifts (institution_id, shift_id)
VALUES (:institution_id, :shift_id)
ON DUPLICATE KEY UPDATE institution_id = institution_id";

$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function remove($institutionId, $shiftId) {
$sql = "DELETE FROM institution_shifts
WHERE institution_id = :institution_id AND shift_id = :shift_id";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':institution_id' => $institutionId,
':shift_id' => $shiftId
]);
}

public function getByInstitution($institutionId) {
$sql = "SELECT s.id, s.name
FROM shifts s
INNER JOIN institution_shifts ins ON s.id = ins.shift_id
WHERE ins.institution_id = :institution_id
ORDER BY s.name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $institutionId]);
return $stmt->fetchAll();
}

public function getInstitutionShiftIds($institutionId) {
$sql = "SELECT shift_id FROM institution_shifts WHERE institution_id = :institution_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $institutionId]);
return $stmt->fetchAll(PDO::FETCH_COLUMN);
}
}
```

5.6 — Justification.php

models/Justification.php PHP

```php
<?php
class Justification {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function create($data) {
$sql = "INSERT INTO justifications
(attendance_id, student_id, submitted_by, reason, document_path)
VALUES (:attendance_id, :student_id, :submitted_by, :reason, :document_path)";

$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function getPending() {
$sql = "SELECT j.*,
CONCAT(u.last_name, ' ', u.first_name) as student_name,
CONCAT(s.last_name, ' ', s.first_name) as submitted_by_name,
a.date as attendance_date,
sub.name as subject_name,
c.name as course_name
FROM justifications j
INNER JOIN users u ON j.student_id = u.id
INNER JOIN users s ON j.submitted_by = s.id
INNER JOIN attendances a ON j.attendance_id = a.id
INNER JOIN subjects sub ON a.subject_id = sub.id
LEFT JOIN course_students cs ON u.id = cs.student_id
LEFT JOIN courses c ON cs.course_id = c.id
WHERE j.status = 'pendiente'
ORDER BY j.created_at DESC";

$stmt = $this->db->query($sql);
return $stmt->fetchAll();
}

public function getByStudent($studentId) {
$sql = "SELECT j.*,
a.date as attendance_date,
sub.name as subject_name,
j.created_at as submitted_at,
j.updated_at as reviewed_at,
CONCAT(r.last_name, ' ', r.first_name) as reviewer_name
FROM justifications j
INNER JOIN attendances a ON j.attendance_id = a.id
INNER JOIN subjects sub ON a.subject_id = sub.id
LEFT JOIN users r ON j.reviewed_by = r.id
WHERE j.student_id = :student_id
ORDER BY j.created_at DESC";

$stmt = $this->db->prepare($sql);
$stmt->execute([':student_id' => $studentId]);
return $stmt->fetchAll();
}

public function approve($justificationId, $reviewerId, $notes = '') {
$sql = "UPDATE justifications
SET status = 'aprobado',
```

```
reviewed_by = :reviewed_by,
review_notes = :notes,
updated_at = NOW()
WHERE id = :id";

$stmt = $this->db->prepare($sql);
$result = $stmt->execute([
':reviewed_by' => $reviewerId,
':notes' => $notes,
':id' => $justificationId
]);

if ($result) {
// Actualizar asistencia a justificado
$sql2 = "UPDATE attendances a
INNER JOIN justifications j ON a.id = j.attendance_id
SET a.status = 'justificado'
WHERE j.id = :id";
$stmt2 = $this->db->prepare($sql2);
$stmt2->execute([':id' => $justificationId]);
}

return $result;
}

public function reject($justificationId, $reviewerId, $notes) {
$sql = "UPDATE justifications
SET status = 'rechazado',
reviewed_by = :reviewed_by,
review_notes = :notes,
updated_at = NOW()
WHERE id = :id";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':reviewed_by' => $reviewerId,
':notes' => $notes,
':id' => $justificationId
]);
}

public function existsByAttendance($attendanceId) {
$sql = "SELECT COUNT(*) as count FROM justifications WHERE attendance_id = :attendance_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':attendance_id' => $attendanceId]);
$result = $stmt->fetch();
return $result['count'] > 0;
}

public function getById($id) {
$sql = "SELECT j.*, j.student_id, j.submitted_by,
CONCAT(u.last_name, ' ', u.first_name) as student_name
FROM justifications j
INNER JOIN users u ON j.student_id = u.id
WHERE j.id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);
return $stmt->fetch();
}

```

```
public function getReviewed($institutionId = null) {
$sql = "SELECT j.*,
CONCAT(u.last_name, ' ', u.first_name) as student_name,
CONCAT(s.last_name, ' ', s.first_name) as submitted_by_name,
CONCAT(r.last_name, ' ', r.first_name) as reviewer_name,
a.date as attendance_date,
a.hour_period,
sub.name as subject_name,
c.name as course_name
FROM justifications j
INNER JOIN users u ON j.student_id = u.id
INNER JOIN users s ON j.submitted_by = s.id
INNER JOIN attendances a ON j.attendance_id = a.id
INNER JOIN subjects sub ON a.subject_id = sub.id
LEFT JOIN users r ON j.reviewed_by = r.id
LEFT JOIN course_students cs ON u.id = cs.student_id
LEFT JOIN courses c ON cs.course_id = c.id
WHERE j.status IN ('aprobado', 'rechazado')
ORDER BY j.updated_at DESC";

$stmt = $this->db->query($sql);
return $stmt->fetchAll();
}

// ■■ Crear justificación por rango de fechas ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
public function createByRange($data) {
$sql = "INSERT INTO justifications
(student_id, submitted_by, date_from, date_to,
working_days, reason_type, reason, document_path, can_approve, status)
VALUES
(:student_id, :submitted_by, :date_from, :date_to,
:working_days, :reason_type, :reason, :document_path, :can_approve, 'pendiente')";
$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

// ■■ Contar días laborables entre dos fechas (lun-vie) ■■■■■■■■■■■■■■■■■■
public static function countWorkingDays($from, $to) {
$count = 0;
$cur = strtotime($from);
$end = strtotime($to);
while ($cur <= $end) {
$dow = (int)date('N', $cur); // 1=lun, 7=dom
if ($dow < 6) $count++; // lun-vie
$cur = strtotime('+1 day', $cur);
}
return $count;
}

// ■■ Determinar quién puede aprobar según días laborables ■■■■■■■■■■■■■■■
// Tutor: hasta 3 días | Inspector/Autoridad: más de 3
public static function resolveApprover($workingDays) {
return $workingDays <= 3 ? 'tutor' : 'inspector';
}

// ■■ Obtener justificaciones pendientes con datos de rango ■■■■■■■■■■■■■■
public function getPendingByApprover($approverRole) {
// tutor ve las suyas (can_approve='tutor')
// inspector/autoridad ven las de inspector y autoridad
if ($approverRole === 'tutor') {
```

```
$where = "j.can_approve = 'tutor'";
} else {
$where = "j.can_approve IN ('inspector','autoridad')";
}

$sql = "SELECT j.*,
CONCAT(u.last_name, ' ', u.first_name) as student_name,
CONCAT(s.last_name, ' ', s.first_name) as submitted_by_name,
a.date as attendance_date,
sub.name as subject_name,
c.name as course_name
FROM justifications j
INNER JOIN users u ON j.student_id = u.id
INNER JOIN users s ON j.submitted_by = s.id
LEFT JOIN attendances a ON j.attendance_id = a.id
LEFT JOIN subjects sub ON a.subject_id = sub.id
LEFT JOIN course_students cs ON u.id = cs.student_id
LEFT JOIN courses c ON cs.course_id = c.id
WHERE j.status = 'pendiente' AND $where
ORDER BY j.created_at DESC";

return $this->db->query($sql)->fetchAll();
}

// ■■ El tutor solo aprueba las de SU curso ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
public function getPendingForTutor($tutorCourseId) {
$sql = "SELECT j.*,
CONCAT(u.last_name, ' ', u.first_name) as student_name,
CONCAT(s.last_name, ' ', s.first_name) as submitted_by_name,
c.name as course_name
FROM justifications j
INNER JOIN users u ON j.student_id = u.id
INNER JOIN users s ON j.submitted_by = s.id
INNER JOIN course_students cs ON u.id = cs.student_id
INNER JOIN courses c ON cs.course_id = c.id
WHERE j.status = 'pendiente'
AND j.can_approve = 'tutor'
AND cs.course_id = :course_id
ORDER BY j.created_at DESC";
$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $tutorCourseId]);
return $stmt->fetchAll();
}

// ■■ Aprobar justificación por rango (sin attendance_id) ■■■■■■■■■■■■■■■
public function approveRange($justificationId, $reviewerId, $notes = '') {
$sql = "UPDATE justifications
SET status='aprobado', reviewed_by=:reviewed_by,
review_notes=:notes, updated_at=NOW()
WHERE id=:id";
$stmt = $this->db->prepare($sql);
$result = $stmt->execute([
':reviewed_by' => $reviewerId,
':notes' => $notes,
':id' => $justificationId,
]);

if ($result) {
// Si tiene attendance_id único, actualizar también esa asistencia
$j = $this->getById($justificationId);
```

```php
if (!empty($j['attendance_id'])) {
$this->db->prepare(
"UPDATE attendances SET status='justificado' WHERE id=:id"
)->execute([':id' => $j['attendance_id']]);
}
// Si tiene rango de fechas, actualizar todas las ausencias del estudiante en ese rango
if (!empty($j['date_from']) && !empty($j['date_to'])) {
$this->db->prepare(
"UPDATE attendances SET status='justificado'
WHERE student_id=:sid AND date BETWEEN :df AND :dt AND status='ausente'"
)->execute([
':sid' => $j['student_id'],
':df' => $j['date_from'],
':dt' => $j['date_to'],
]);
}
}
return $result;
}

// Crear una justificación por cada asistencia seleccionada
public function createForAttendances($attendanceIds, $data) {
$sql = "INSERT INTO justifications
(attendance_id, student_id, submitted_by,
date_from, date_to, working_days,
reason_type, reason, document_path, can_approve, status)
VALUES
(:attendance_id, :student_id, :submitted_by,
:date_from, :date_to, :working_days,
:reason_type, :reason, :document_path, :can_approve, 'pendiente')";
$stmt = $this->db->prepare($sql);
$ok = true;
foreach ($attendanceIds as $attId) {
$row = $data;
$row[':attendance_id'] = (int)$attId;
if (!$stmt->execute($row)) $ok = false;
}
return $ok;
}

// Contar ausencias ya justificadas (pendiente o aprobada) para un estudiante en un rango
public function countPendingJustifications($studentId) {
$sql = "SELECT COUNT(*) FROM justifications
WHERE student_id = :sid AND status = 'pendiente'";
$stmt = $this->db->prepare($sql);
$stmt->execute([':sid' => $studentId]);
return (int)$stmt->fetchColumn();
}
}
```

5.7 — Notification.php

models/Notification.php PHP

```php
<?php
class Notification {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

// Crear notificación
// $link: URL relativa destino, ej: '?action=pending_justifications'
public function create($userId, $title, $message, $type = 'info', $link = null) {
$sql = "INSERT INTO notifications (user_id, title, message, type, link, is_read, created_at)
VALUES (:user_id, :title, :message, :type, :link, 0, NOW())";
$stmt = $this->db->prepare($sql);
return $stmt->execute([
':user_id' => $userId,
':title' => $title,
':message' => $message,
':type' => $type,
':link' => $link
]);
}

// Obtener notificaciones de un usuario (con paginación)
public function getByUser($userId, $limit = 20, $offset = 0) {
$sql = "SELECT * FROM notifications
WHERE user_id = :user_id
ORDER BY created_at DESC
LIMIT :limit OFFSET :offset";
$stmt = $this->db->prepare($sql);
$stmt->bindValue(':user_id', $userId, PDO::PARAM_INT);
$stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
$stmt->bindValue(':offset', $offset, PDO::PARAM_INT);
$stmt->execute();
return $stmt->fetchAll();
}

// Solo no leídas (para el dropdown del navbar)
public function getUnread($userId, $limit = 10) {
$sql = "SELECT * FROM notifications
WHERE user_id = :user_id AND is_read = 0
ORDER BY created_at DESC
LIMIT :limit";
$stmt = $this->db->prepare($sql);
$stmt->bindValue(':user_id', $userId, PDO::PARAM_INT);
$stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
$stmt->execute();
return $stmt->fetchAll();
}

// Contar no leídas
public function getUnreadCount($userId) {
$sql = "SELECT COUNT(*) as count FROM notifications
WHERE user_id = :user_id AND is_read = 0";
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);
return (int)$stmt->fetch()['count'];
}

// Marcar una como leída
```

```php
public function markAsRead($notificationId, $userId) {
$sql = "UPDATE notifications SET is_read = 1
WHERE id = :id AND user_id = :user_id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $notificationId, ':user_id' => $userId]);
}

// Marcar TODAS como leídas
public function markAllAsRead($userId) {
$sql = "UPDATE notifications SET is_read = 1
WHERE user_id = :user_id AND is_read = 0";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':user_id' => $userId]);
}

// Eliminar una notificación
public function delete($notificationId, $userId) {
$sql = "DELETE FROM notifications WHERE id = :id AND user_id = :user_id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $notificationId, ':user_id' => $userId]);
}

// Eliminar todas las leídas
public function deleteRead($userId) {
$sql = "DELETE FROM notifications WHERE user_id = :user_id AND is_read = 1";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':user_id' => $userId]);
}

// Total de notificaciones del usuario (para paginación)
public function countByUser($userId) {
$sql = "SELECT COUNT(*) as count FROM notifications WHERE user_id = :user_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);
return (int)$stmt->fetch()['count'];
}
}
```

5.8 — Representative.php

models/Representative.php PHP

```php
<?php
class Representative {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function assignStudent($representativeId, $studentId, $relationship, $isPrimary = 0) {
$sql = "INSERT INTO representatives
(representative_id, student_id, relationship, is_primary)
VALUES
(:rep_id, :stu_id, :rel_ins, :prim_ins)
ON DUPLICATE KEY UPDATE
relationship = :rel_upd,
is_primary = :prim_upd";

$stmt = $this->db->prepare($sql);

return $stmt->execute([
':rep_id' => $representativeId,
':stu_id' => $studentId,
':rel_ins' => $relationship,
':prim_ins' => $isPrimary,
':rel_upd' => $relationship,
':prim_upd' => $isPrimary
]);
}

public function getStudentsByRepresentative($representativeId) {
$sql = "SELECT u.*, r.relationship, r.is_primary, c.name as course_name, sh.name as shift_name
FROM users u
INNER JOIN representatives r ON u.id = r.student_id
LEFT JOIN course_students cs ON u.id = cs.student_id
LEFT JOIN courses c ON cs.course_id = c.id
LEFT JOIN shifts sh ON c.shift_id = sh.id
WHERE r.representative_id = :representative_id
ORDER BY r.is_primary DESC, u.last_name, u.first_name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':representative_id' => $representativeId]);
return $stmt->fetchAll();
}

public function getRepresentativesByStudent($studentId) {
$sql = "SELECT u.*, r.relationship, r.is_primary
FROM users u
INNER JOIN representatives r ON u.id = r.representative_id
WHERE r.student_id = :student_id
ORDER BY r.is_primary DESC";

$stmt = $this->db->prepare($sql);
$stmt->execute([':student_id' => $studentId]);
return $stmt->fetchAll();
}

public function removeStudent($representativeId, $studentId) {
$sql = "DELETE FROM representatives
WHERE representative_id = :representative_id AND student_id = :student_id";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':representative_id' => $representativeId,
':student_id' => $studentId
]);
}
}
```

5.9 — Role.php

models/Role.php PHP

```php
<?php
class Role {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function getAll() {
$sql = "SELECT * FROM roles ORDER BY name";
$stmt = $this->db->query($sql);
return $stmt->fetchAll();
}

public function findById($id) {
$sql = "SELECT * FROM roles WHERE id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);
return $stmt->fetch();
}
}
```

5.10 — SchoolYear.php

models/SchoolYear.php PHP

```php
<?php
class SchoolYear {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function getActive() {
$sql = "SELECT * FROM school_years
WHERE institution_id = :institution_id AND is_active = 1
LIMIT 1";

$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $_SESSION['institution_id']]);
return $stmt->fetch();
}

public function getAll() {
$sql = "SELECT * FROM school_years
WHERE institution_id = :institution_id
ORDER BY start_date DESC";

$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $_SESSION['institution_id']]);
return $stmt->fetchAll();
}


public function findById($id) {
$sql = "SELECT * FROM school_years
WHERE id = :id AND institution_id = :institution_id";

$stmt = $this->db->prepare($sql);
$stmt->execute([
':id' => $id,
':institution_id' => $_SESSION['institution_id']
]);

return $stmt->fetch();
}

public function create($data) {
$sql = "INSERT INTO school_years
(institution_id, name, start_date, end_date, is_active)
VALUES
(:institution_id, :name, :start_date, :end_date, :is_active)";

$stmt = $this->db->prepare($sql);
$ok = $stmt->execute([
':institution_id' => $data['institution_id'],
':name' => $data['name'],
':start_date' => $data['start_date'],
':end_date' => $data['end_date'],
':is_active' => $data['is_active']
]);
// Retorna el ID insertado para poder activarlo en la misma conexión
return $ok ? (int)$this->db->lastInsertId() : 0;
}

```

```
public function update($data) {
$sql = "UPDATE school_years
SET name=:name,
start_date=:start_date,
end_date=:end_date
WHERE id=:id
AND institution_id=:institution_id";

$stmt = $this->db->prepare($sql);

return $stmt->execute([
':id'=>$data['id'],
':name'=>$data['name'],
':start_date'=>$data['start_date'],
':end_date'=>$data['end_date'],
':institution_id'=>$_SESSION['institution_id']
]);
}

public function delete($id) {
$stmt = $this->db->prepare(
"DELETE FROM school_years
WHERE id=:id AND institution_id=:institution_id"
);

return $stmt->execute([
':id'=>$id,
':institution_id'=>$_SESSION['institution_id']
]);
}

public function activate($id) {

// desactiva todos
$this->db->prepare(
"UPDATE school_years
SET is_active=0
WHERE institution_id=:institution_id"
)->execute([':institution_id'=>$_SESSION['institution_id']]);

if ($id == 0) return true;

// activa uno
$stmt = $this->db->prepare(
"UPDATE school_years
SET is_active=1
WHERE id=:id AND institution_id=:institution_id"
);

return $stmt->execute([
':id'=>$id,
':institution_id'=>$_SESSION['institution_id']
]);
}

public function deactivate($id) {
$stmt = $this->db->prepare(
"UPDATE school_years
SET is_active=0
WHERE id=:id AND institution_id=:institution_id"
```

```
);

return $stmt->execute([
':id'=>$id,
':institution_id'=>$_SESSION['institution_id']
]);
}

public function checkOverlap($start, $end, $excludeId = null) {

$sql = "SELECT COUNT(*)
FROM school_years
WHERE institution_id = :institution_id
AND (start_date <= :end AND end_date >= :start)";

if ($excludeId) {
$sql .= " AND id != :excludeId";
}

$stmt = $this->db->prepare($sql);

$params = [
':institution_id' => $_SESSION['institution_id'],
':start' => $start,
':end' => $end
];

if ($excludeId) {
$params[':excludeId'] = $excludeId;
}

$stmt->execute($params);
return $stmt->fetchColumn() > 0;
}

}
```

5.11 — Shift.php

models/Shift.php PHP

```
<?php
class Shift {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function getAll() {
$sql = "SELECT * FROM shifts ORDER BY id";
$stmt = $this->db->query($sql);
return $stmt->fetchAll();
}
}
```

5.12 — Subject.php

models/Subject.php PHP

```php
<?php
class Subject {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function getAll() {
$sql = "SELECT * FROM subjects
WHERE institution_id = :institution_id
ORDER BY name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':institution_id' => $_SESSION['institution_id']]);
return $stmt->fetchAll();
}

public function create($data) {
$sql = "INSERT INTO subjects (institution_id, name, code)
VALUES (:institution_id, :name, :code)";

$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function findById($id) {
$sql = "SELECT * FROM subjects WHERE id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);
return $stmt->fetch();
}

public function update($data) {
$sql = "UPDATE subjects SET
name = :name,
code = :code,
updated_at = NOW()
WHERE id = :id";

$stmt = $this->db->prepare($sql);
return $stmt->execute($data);
}

public function delete($id) {
$sql = "DELETE FROM subjects WHERE id = :id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $id]);
}
}
```

5.13 — TeacherAssignment.php

models/TeacherAssignment.php PHP

```php
<?php
class TeacherAssignment {
private $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function assign($data) {
// Verificar si ya existe asignación curso-materia
$checkSql = "SELECT ta.id, CONCAT(u.last_name, ' ', u.first_name) as teacher_name
FROM teacher_assignments ta
INNER JOIN users u ON ta.teacher_id = u.id
WHERE ta.course_id = :course_id
AND ta.subject_id = :subject_id
AND ta.school_year_id = :school_year_id";

$checkStmt = $this->db->prepare($checkSql);
$checkStmt->execute([
':course_id' => $data[':course_id'],
':subject_id' => $data[':subject_id'],
':school_year_id' => $data[':school_year_id']
]);

$existing = $checkStmt->fetch();

if ($existing) {
return [
'success' => false,
'message' => 'La materia ya está asignada a: ' . $existing['teacher_name']
];
}

$sql = "INSERT INTO teacher_assignments
(teacher_id, course_id, subject_id, school_year_id, is_tutor)
VALUES (:teacher_id, :course_id, :subject_id, :school_year_id, :is_tutor)";

$stmt = $this->db->prepare($sql);
$result = $stmt->execute($data);

return [
'success' => $result,
'message' => 'Asignación creada correctamente'
];
}

public function getByTeacher($teacherId) {
$sql = "SELECT ta.*, c.name as course_name, s.name as subject_name,
sh.name as shift_name, sy.name as year_name
FROM teacher_assignments ta
INNER JOIN courses c ON ta.course_id = c.id
INNER JOIN subjects s ON ta.subject_id = s.id
INNER JOIN shifts sh ON c.shift_id = sh.id
INNER JOIN school_years sy ON ta.school_year_id = sy.id
WHERE ta.teacher_id = :teacher_id
ORDER BY c.name, s.name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':teacher_id' => $teacherId]);
return $stmt->fetchAll();
```

```
}

public function getByCourse($courseId) {
$sql = "SELECT ta.*,
CONCAT(u.last_name, ' ', u.first_name) as teacher_name,
s.name as subject_name
FROM teacher_assignments ta
INNER JOIN users u ON ta.teacher_id = u.id
INNER JOIN subjects s ON ta.subject_id = s.id
WHERE ta.course_id = :course_id
ORDER BY s.name";

$stmt = $this->db->prepare($sql);
$stmt->execute([':course_id' => $courseId]);
return $stmt->fetchAll();
}

public function getTutorByCourse($courseId, $schoolYearId) {
$sql = "SELECT u.*, ta.id as assignment_id
FROM teacher_assignments ta
INNER JOIN users u ON ta.teacher_id = u.id
WHERE ta.course_id = :course_id
AND ta.school_year_id = :school_year_id
AND ta.is_tutor = 1
LIMIT 1";

$stmt = $this->db->prepare($sql);
$stmt->execute([
':course_id' => $courseId,
':school_year_id' => $schoolYearId
]);
return $stmt->fetch();
}

public function setTutor($courseId, $teacherId, $schoolYearId) {
// Verificar si el docente ya es tutor de otro curso en el mismo año lectivo
$checkSql = "SELECT c.name
FROM teacher_assignments ta
INNER JOIN courses c ON ta.course_id = c.id
WHERE ta.teacher_id = :teacher_id
AND ta.school_year_id = :school_year_id
AND ta.is_tutor = 1
AND ta.course_id != :course_id";

$checkStmt = $this->db->prepare($checkSql);
$checkStmt->execute([
':teacher_id' => $teacherId,
':school_year_id' => $schoolYearId,
':course_id' => $courseId
]);

$existingTutor = $checkStmt->fetch();

if ($existingTutor) {
return [
'success' => false,
'message' => 'Este docente ya es tutor del curso: ' . $existingTutor['name']
];
}

```

```
// Quitar tutor actual del curso
$sql1 = "UPDATE teacher_assignments
SET is_tutor = 0
WHERE course_id = :course_id AND school_year_id = :school_year_id";
$stmt1 = $this->db->prepare($sql1);
$stmt1->execute([
':course_id' => $courseId,
':school_year_id' => $schoolYearId
]);

// Verificar si docente tiene asignación en el curso
$checkAssignment = "SELECT id FROM teacher_assignments
WHERE course_id = :course_id
AND teacher_id = :teacher_id
AND school_year_id = :school_year_id
LIMIT 1";
$stmt = $this->db->prepare($checkAssignment);
$stmt->execute([
':course_id' => $courseId,
':teacher_id' => $teacherId,
':school_year_id' => $schoolYearId
]);
$existing = $stmt->fetch();

if ($existing) {
// Actualizar asignación existente
$sql2 = "UPDATE teacher_assignments
SET is_tutor = 1
WHERE id = :id";
$stmt2 = $this->db->prepare($sql2);
$result = $stmt2->execute([':id' => $existing['id']]);
} else {
return [
'success' => false,
'message' => 'El docente debe tener al menos una asignatura en el curso antes de ser tutor'
];
}

return [
'success' => $result,
'message' => 'Tutor asignado correctamente'
];
}

public function remove($assignmentId) {
// Verificar si es tutor antes de eliminar
$checkSql = "SELECT is_tutor FROM teacher_assignments WHERE id = :id";
$checkStmt = $this->db->prepare($checkSql);
$checkStmt->execute([':id' => $assignmentId]);
$assignment = $checkStmt->fetch();

if ($assignment && $assignment['is_tutor'] == 1) {
return [
'success' => false,
'message' => 'No se puede eliminar: este docente es tutor del curso. Quite primero la tutoría.'
];
}

$sql = "DELETE FROM teacher_assignments WHERE id = :id";
$stmt = $this->db->prepare($sql);
```

```
$result = $stmt->execute([':id' => $assignmentId]);

return [
'success' => $result,
'message' => 'Asignación eliminada correctamente'
];
}

public function getAll() {
$sql = "SELECT ta.*,
CONCAT(u.last_name, ' ', u.first_name) as teacher_name,
c.name as course_name,
s.name as subject_name,
sy.name as year_name
FROM teacher_assignments ta
INNER JOIN users u ON ta.teacher_id = u.id
INNER JOIN courses c ON ta.course_id = c.id
INNER JOIN subjects s ON ta.subject_id = s.id
INNER JOIN school_years sy ON ta.school_year_id = sy.id
ORDER BY c.name, s.name";

$stmt = $this->db->query($sql);
return $stmt->fetchAll();
}
}
```

5.14 — User.php

models/User.php PHP

```php
<?php
class User {
public $db;

public function __construct($database) {
$this->db = $database->connect();
}

public function create($data) {
$sql = "INSERT INTO users (institution_id, username, email, password, first_name, last_name, dni, phone)
VALUES (:institution_id, :username, :email, :password, :first_name, :last_name, :dni, :phone)";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':institution_id' => $data['institution_id'],
':username' => $data['username'],
':email' => $data['email'],
':password' => Security::hashPassword($data['password']),
':first_name' => $data['first_name'],
':last_name' => $data['last_name'],
':dni' => $data['dni'] ?? null,
':phone' => $data['phone'] ?? null
]);
}

public function findByEmail($email) {
$sql = "SELECT * FROM users WHERE email = :email AND is_active = 1";
$stmt = $this->db->prepare($sql);
$stmt->execute([':email' => $email]);
return $stmt->fetch();
}

public function findById($id) {
$sql = "SELECT * FROM users WHERE id = :id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':id' => $id]);
return $stmt->fetch();
}

public function getUserRoles($userId) {
$sql = "SELECT r.name FROM roles r
INNER JOIN user_roles ur ON r.id = ur.role_id
WHERE ur.user_id = :user_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);
return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

public function assignRole($userId, $roleId) {
$sql = "INSERT IGNORE INTO user_roles (user_id, role_id) VALUES (:user_id, :role_id)";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':user_id' => $userId, ':role_id' => $roleId]);
}

public function setResetToken($email, $token) {
$sql = "UPDATE users SET reset_token = :token, reset_expires = DATE_ADD(NOW(), INTERVAL 1 HOUR)
WHERE email = :email";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':token' => $token, ':email' => $email]);
}
```

```
61
62 public function validateResetToken($token) {
63 $sql = "SELECT * FROM users WHERE reset_token = :token AND reset_expires > NOW()";
64 $stmt = $this->db->prepare($sql);
65 $stmt->execute([':token' => $token]);
66 return $stmt->fetch();
67 }
68
69 public function resetPassword($userId, $newPassword) {
70 $sql = "UPDATE users SET password = :password, reset_token = NULL, reset_expires = NULL
71 WHERE id = :id";
72 $stmt = $this->db->prepare($sql);
73 return $stmt->execute([
74 ':password' => Security::hashPassword($newPassword),
75 ':id' => $userId
76 ]);
77 }
78
79 public function getAll() {
80 $sql = "SELECT u.*, GROUP_CONCAT(r.name) as roles
81 FROM users u
82 LEFT JOIN user_roles ur ON u.id = ur.user_id
83 LEFT JOIN roles r ON ur.role_id = r.id
84 WHERE u.institution_id = :institution_id
85 GROUP BY u.id
86 ORDER BY u.last_name, u.first_name";
87
88 $stmt = $this->db->prepare($sql);
89 $stmt->execute([':institution_id' => $_SESSION['institution_id']]);
90 return $stmt->fetchAll();
91 }
92
93 public function removeRole($userId, $roleId) {
94 $sql = "DELETE FROM user_roles WHERE user_id = :user_id AND role_id = :role_id";
95 $stmt = $this->db->prepare($sql);
96 return $stmt->execute([':user_id' => $userId, ':role_id' => $roleId]);
97 }
98
99 public function getUserWithRoles($userId) {
100 $user = $this->findById($userId);
101 if ($user) {
102 $user['roles'] = $this->getUserRoles($userId);
103 }
104 return $user;
105 }
106
107 /* public function getByRole($roleName) {
108 $sql = "SELECT u.* FROM users u
109 INNER JOIN user_roles ur ON u.id = ur.user_id
110 INNER JOIN roles r ON ur.role_id = r.id
111 WHERE r.name = :role_name AND u.institution_id = :institution_id
112 ORDER BY u.last_name, u.first_name";
113
114 $stmt = $this->db->prepare($sql);
115 $stmt->execute([
116 ':role_name' => $roleName,
117 ':institution_id' => $_SESSION['institution_id']
118 ]);
119 return $stmt->fetchAll();
120 } */
```

```
121
122 // Agregar al final de la clase User
123
124 public function getByRole($roleName) {
125 $sql = "SELECT u.* FROM users u
126 INNER JOIN user_roles ur ON u.id = ur.user_id
127 INNER JOIN roles r ON ur.role_id = r.id
128 WHERE r.name = :role_name AND u.institution_id = :institution_id
129 ORDER BY u.last_name, u.first_name";
130
131 $stmt = $this->db->prepare($sql);
132 $stmt->execute([
133 ':role_name' => $roleName,
134 ':institution_id' => $_SESSION['institution_id']
135 ]);
136 return $stmt->fetchAll();
137 }
138
139 public function getStudentsNotEnrolled($schoolYearId) {
140 $sql = "SELECT u.* FROM users u
141 INNER JOIN user_roles ur ON u.id = ur.user_id
142 INNER JOIN roles r ON ur.role_id = r.id
143 WHERE r.name = 'estudiante'
144 AND u.institution_id = :institution_id
145 AND u.id NOT IN (
146 SELECT student_id FROM course_students
147 WHERE school_year_id = :school_year_id
148 )
149 ORDER BY u.last_name, u.first_name";
150
151 $stmt = $this->db->prepare($sql);
152 $stmt->execute([
153 ':institution_id' => $_SESSION['institution_id'],
154 ':school_year_id' => $schoolYearId
155 ]);
156 return $stmt->fetchAll();
157 }
158
159 public function getStudentCourse($studentId, $schoolYearId) {
160 $sql = "SELECT c.*, s.name as shift_name
161 FROM courses c
162 INNER JOIN course_students cs ON c.id = cs.course_id
163 INNER JOIN shifts s ON c.shift_id = s.id
164 WHERE cs.student_id = :student_id
165 AND cs.school_year_id = :school_year_id";
166
167 $stmt = $this->db->prepare($sql);
168 $stmt->execute([
169 ':student_id' => $studentId,
170 ':school_year_id' => $schoolYearId
171 ]);
172 return $stmt->fetch();
173 }
174
175 public function updateProfile($userId, $data) {
176 $sql = "UPDATE users
177 SET first_name = :first_name,
178 last_name = :last_name,
179 phone = :phone,
180 email = :email,
```

```
updated_at = NOW()
WHERE id = :id";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':first_name' => $data['first_name'],
':last_name' => $data['last_name'],
':phone' => $data['phone'],
':email' => $data['email'],
':id' => $userId
]);
}

public function updatePassword($userId, $newPassword) {
$sql = "UPDATE users SET password = :password WHERE id = :id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([
':password' => Security::hashPassword($newPassword),
':id' => $userId
]);
}

public function findByEmailOrUsername($emailOrUsername) {
$sql = "SELECT * FROM users
WHERE (email = :email OR username = :username)
AND is_active = 1";

$stmt = $this->db->prepare($sql);
$stmt->execute([
':email' => $emailOrUsername,
':username' => $emailOrUsername
]);
return $stmt->fetch();
}

public function getUserRoleIds($userId) {
$sql = "SELECT r.id FROM roles r
INNER JOIN user_roles ur ON r.id = ur.role_id
WHERE ur.user_id = :user_id
ORDER BY r.name";
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);
return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

public function findByUsername($username) {
$sql = "SELECT * FROM users WHERE username = :username";
$stmt = $this->db->prepare($sql);
$stmt->execute([':username' => $username]);
return $stmt->fetch();
}

public function update($data) {
// Si se incluye password, actualizarla
if (isset($data['password'])) {
$sql = "UPDATE users SET
username = :username,
email = :email,
password = :password,
first_name = :first_name,
```

```
last_name = :last_name,
dni = :dni,
phone = :phone,
updated_at = NOW()
WHERE id = :id";

$params = [
':id' => $data['id'],
':username' => $data['username'],
':email' => $data['email'],
':password' => Security::hashPassword($data['password']),
':first_name' => $data['first_name'],
':last_name' => $data['last_name'],
':dni' => $data['dni'],
':phone' => $data['phone']
];
} else {
// No actualizar password
$sql = "UPDATE users SET
username = :username,
email = :email,
first_name = :first_name,
last_name = :last_name,
dni = :dni,
phone = :phone,
updated_at = NOW()
WHERE id = :id";

$params = [
':id' => $data['id'],
':username' => $data['username'],
':email' => $data['email'],
':first_name' => $data['first_name'],
':last_name' => $data['last_name'],
':dni' => $data['dni'],
':phone' => $data['phone']
];
}

$stmt = $this->db->prepare($sql);
return $stmt->execute($params);
}

public function delete($userId) {
// Primero eliminar roles
$sql = "DELETE FROM user_roles WHERE user_id = :user_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);

// Eliminar representaciones (tanto como representante como estudiante)
$sql = "DELETE FROM representatives
WHERE representative_id = :rep_id OR student_id = :stu_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([
':rep_id' => $userId,
':stu_id' => $userId
]);

// Eliminar de course_students
$sql = "DELETE FROM course_students WHERE student_id = :user_id";
```

```
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);

// Eliminar asignaciones docentes
$sql = "DELETE FROM teacher_assignments WHERE teacher_id = :user_id";
$stmt = $this->db->prepare($sql);
$stmt->execute([':user_id' => $userId]);

// Finalmente eliminar usuario
$sql = "DELETE FROM users WHERE id = :id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $userId]);
}

public function deactivate($userId) {
$sql = "UPDATE users SET is_active = 0, updated_at = NOW() WHERE id = :id";
$stmt = $this->db->prepare($sql);
return $stmt->execute([':id' => $userId]);
}
}
```

6. MODULO HELPERS (4 archivos)

6.1 — Backup.php

helpers/Backup.php PHP

```php
<?php
class Backup {
private $dbHost = 'localhost';
private $dbUser = 'root';
private $dbPass = '';
private $dbName = 'ecuasistencia2026_db';
private $backupDir;

public function __construct() {
$this->backupDir = BASE_PATH . '/backups/';
if (!file_exists($this->backupDir)) {
mkdir($this->backupDir, 0755, true);
}
}

public function createBackup() {
$filename = 'backup_' . date('Y-m-d_H-i-s') . '.sql';
$filepath = $this->backupDir . $filename;

// Verificar que mysqldump esté disponible
$mysqldumpPath = 'C:\\xampp\\mysql\\bin\\mysqldump.exe'; // XAMPP Windows
if (!file_exists($mysqldumpPath)) {
$mysqldumpPath = 'mysqldump'; // Linux/Mac o PATH configurado
}

// Comando mejorado con opciones
if (empty($this->dbPass)) {
$command = sprintf(
'"%s" --host=%s --user=%s --skip-password %s > "%s" 2>&1',
$mysqldumpPath,
$this->dbHost,
$this->dbUser,
$this->dbName,
$filepath
);
} else {
$command = sprintf(
'"%s" --host=%s --user=%s --password=%s %s > "%s" 2>&1',
$mysqldumpPath,
$this->dbHost,
$this->dbUser,
$this->dbPass,
$this->dbName,
$filepath
);
}

exec($command, $output, $result);

// Verificar que el archivo existe y tiene contenido
if (file_exists($filepath) && filesize($filepath) > 0) {
return $filename;
}

// Si falló, intentar eliminar archivo vacío
if (file_exists($filepath)) {
unlink($filepath);
}

return false;
```

```
}

public function getBackups() {
$files = glob($this->backupDir . 'backup_*.sql');
$backups = [];

foreach ($files as $file) {
$size = filesize($file);
$backups[] = [
'filename' => basename($file),
'path' => $file,
'size' => $this->formatBytes($size),
'date' => date('d/m/Y H:i:s', filemtime($file)),
'timestamp' => filemtime($file)
];
}

usort($backups, function($a, $b) {
return $b['timestamp'] - $a['timestamp'];
});

return $backups;
}

private function formatBytes($bytes, $precision = 2) {
$units = ['B', 'KB', 'MB', 'GB'];
$bytes = max($bytes, 0);
$pow = floor(($bytes ? log($bytes) : 0) / log(1024));
$pow = min($pow, count($units) - 1);
$bytes /= (1 << (10 * $pow));
return round($bytes, $precision) . ' ' . $units[$pow];
}

public function deleteOldBackups($daysToKeep = 30) {
$files = glob($this->backupDir . 'backup_*.sql');
$deleted = 0;

foreach ($files as $file) {
if (time() - filemtime($file) > ($daysToKeep * 24 * 60 * 60)) {
unlink($file);
$deleted++;
}
}

return $deleted;
}

public function downloadBackup($filename) {
$filepath = $this->backupDir . $filename;

if (!file_exists($filepath)) {
return false;
}

header('Content-Type: application/octet-stream');
header('Content-Disposition: attachment; filename="' . $filename . '"');
header('Content-Length: ' . filesize($filepath));
readfile($filepath);
exit;
}
```

```
121
122 public function deleteBackup($filename) {
123 // Validar que sea un archivo de backup válido
124 if (!preg_match('/^backup_[\d\-_]+\.sql$/', $filename)) {
125 return false;
126 }
127
128 $filepath = $this->backupDir . $filename;
129
130 if (file_exists($filepath)) {
131 return unlink($filepath);
132 }
133
134 return false;
135 }
136 }
```

6.2 — Logger.php

helpers/Logger.php PHP

```php
<?php
class Logger {
private $db;

public function __construct() {
$database = new Database();
$this->db = $database->connect();
}

public function log($action, $entityType = null, $entityId = null, $description = null) {
if (!isset($_SESSION['user_id'])) {
return false;
}

$sql = "INSERT INTO activity_logs
(user_id, action, entity_type, entity_id, description, ip_address, user_agent)
VALUES (:user_id, :action, :entity_type, :entity_id, :description, :ip_address, :user_agent)";

$stmt = $this->db->prepare($sql);
return $stmt->execute([
':user_id' => $_SESSION['user_id'],
':action' => $action,
':entity_type' => $entityType,
':entity_id' => $entityId,
':description' => $description,
':ip_address' => $_SERVER['REMOTE_ADDR'] ?? null,
':user_agent' => $_SERVER['HTTP_USER_AGENT'] ?? null
]);
}

public function getLogs($limit = 100, $filters = []) {
$sql = "SELECT l.*, CONCAT(u.last_name, ' ', u.first_name) as user_name
FROM activity_logs l
INNER JOIN users u ON l.user_id = u.id
WHERE 1=1";

$params = [];

if (!empty($filters['user_id'])) {
$sql .= " AND l.user_id = :user_id";
$params[':user_id'] = $filters['user_id'];
}

if (!empty($filters['action'])) {
$sql .= " AND l.action = :action";
$params[':action'] = $filters['action'];
}

if (!empty($filters['start_date'])) {
$sql .= " AND l.created_at >= :start_date";
$params[':start_date'] = $filters['start_date'];
}

if (!empty($filters['end_date'])) {
$sql .= " AND l.created_at <= :end_date";
$params[':end_date'] = $filters['end_date'] . ' 23:59:59';
}

$sql .= " ORDER BY l.created_at DESC LIMIT :limit";

```

```
$stmt = $this->db->prepare($sql);
foreach ($params as $key => $value) {
$stmt->bindValue($key, $value);
}
$stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
$stmt->execute();

return $stmt->fetchAll();
}
}
```

6.3 — Mailer.php

helpers/Mailer.php PHP

```
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require BASE_PATH . '/vendor/autoload.php';

class Mailer {
public static function send($to, $subject, $body) {
$mail = new PHPMailer(true);

try {
$mail->isSMTP();
$mail->Host = SMTP_HOST;
$mail->SMTPAuth = true;
$mail->Username = SMTP_USER;
$mail->Password = SMTP_PASS;
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->Port = SMTP_PORT;
$mail->CharSet = 'UTF-8';

$mail->setFrom(SMTP_FROM, SMTP_NAME);
$mail->addAddress($to);

$mail->isHTML(true);
$mail->Subject = $subject;
$mail->Body = $body;

$mail->send();
return true;
} catch (Exception $e) {
error_log("Error de correo: {$mail->ErrorInfo}");
return false;
}
}
}
```

6.4 — Security.php

helpers/Security.php PHP

```php
<?php
class Security {
public static function generateToken() {
if (!isset($_SESSION['csrf_token'])) {
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
return $_SESSION['csrf_token'];
}

public static function validateToken($token) {
return isset($_SESSION['csrf_token']) && hash_equals($_SESSION['csrf_token'], $token);
}

public static function sanitize($data) {
return htmlspecialchars(strip_tags(trim($data)), ENT_QUOTES, 'UTF-8');
}

public static function validateEmail($email) {
return filter_var($email, FILTER_VALIDATE_EMAIL);
}

public static function hashPassword($password) {
return password_hash($password, PASSWORD_BCRYPT);
}

public static function verifyPassword($password, $hash) {
return password_verify($password, $hash);
}

public static function requireLogin() {
if (!isset($_SESSION['user_id'])) {
header('Location: ' . BASE_URL . '/public/index.php?action=login');
exit;
}
}

public static function hasRole($roleNames) {
// Superusuario siempre tiene acceso
if (!empty($_SESSION['is_superadmin'])) return true;
if (!isset($_SESSION['roles'])) return false;
$roleNames = (array) $roleNames;
return count(array_intersect($roleNames, $_SESSION['roles'])) > 0;
}

public static function isSuperAdmin() {
return !empty($_SESSION['is_superadmin']);
}

public static function startSession($rememberMe = false) {
if ($rememberMe) {
// Sesión de 7 días
$lifetime = 7 * 24 * 60 * 60; // 7 días
} else {
// Sesión expira al cerrar navegador
$lifetime = 0;
}

session_set_cookie_params([
'lifetime' => $lifetime,
'path' => '/',
```

```
'httponly' => true,
'samesite' => 'Lax'
]);

session_start();

// Regenerar ID por seguridad
if (!isset($_SESSION['initiated'])) {
session_regenerate_id(true);
$_SESSION['initiated'] = true;
}

// Timeout de inactividad (30 minutos)
if (isset($_SESSION['last_activity']) &&
(time() - $_SESSION['last_activity'] > 1800)) {
session_unset();
session_destroy();
return false;
}

$_SESSION['last_activity'] = time();
return true;
}
}
```