

UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas



Proyecto Unidad III

“Mejoramientos de la Aplicación”

Curso: Calidad y Pruebas de Software

Docente: Ing. Cuadros Quiroga, Patrick Jose

Integrantes

LOPEZ CATUNTA, Brayar Christian	(2021069822)
BRICEÑO DIAZ, Jorge Luis	(2017059611)
CUADROS GARCIA, Mirian	(2021071083)
CASTAÑEDA CENTURION, Jorge Enrique	(2021069822)

**Tacna – Perú
2024**



INDICE GENERAL

I.	INTRODUCCION.....	3
II.	RESUMEN.....	4
III.	OBJETIVOS	5
1.	Objetivo General	5
2.	Objetivos Específicos.....	5
IV.	PLANTEAMIENTO DEL PROBLEMA.....	5
1.	Problema	5
2.	Justificación.....	6
3.	Alcance.....	6
V.	DESARROLLO DEL TRABAJO	7
1.	Diagrama de Caso de Uso	7
2.	Diagrama de Paquetes	8
3.	Diagrama de Clases	8
4.	Crear Diagrama de Componentes	9
5.	Analisis de la aplicación con SonarQube y Snyk	9
6.	Reporte de Cobertura de Pruebas	22
7.	Reporte de Pruebas guiadas por el comportamiento (BDD Given When Then).....	91
8.	Pasos para hacer la Automatización de “Github Action”	99
9.	Cronograma	108



I. INTRODUCCION

El presente trabajo aborda la creación de un sistema automatizado para la gestión de alquiler de vehículos, desarrollado con PHP8 y MySQL. El sistema se basa en la arquitectura MVC (Modelo-Vista-Controlador) para estructurar el código de manera eficiente y mantener una clara separación entre la lógica de negocios, la presentación y el control de la aplicación. Este sistema busca resolver problemas comunes en la gestión de alquiler de vehículos, como errores en las reservas, duplicaciones y una experiencia de usuario deficiente.

El proyecto se centra en la creación de una plataforma en línea que permite a los usuarios registrarse y realizar reservas de vehículos de manera sencilla y segura. Los administradores pueden gestionar el inventario de vehículos, las reservas y los clientes, así como generar reportes y enviar notificaciones por correo electrónico. La aplicación está diseñada para ser intuitiva y permite una fácil personalización para adaptarse a diferentes necesidades empresariales.

El sistema proporcionará una solución centralizada para la gestión de alquiler de vehículos, mejorando la eficiencia operativa y reduciendo los errores humanos. Además, ofrecerá una mejor experiencia al usuario, facilitando el proceso de alquiler y brindando un servicio más confiable y accesible.



II. RESUMEN

El proyecto tiene como objetivo principal desarrollar un sistema automatizado que facilite el proceso de alquiler de vehículos mediante una plataforma en línea. Este sistema permitirá a los usuarios registrarse y realizar reservas de vehículos, mientras que los administradores podrán gestionar el inventario de vehículos, las reservas y los clientes. Además, incluirá la generación de reportes y el envío de notificaciones por correo electrónico.

El sistema está diseñado para ser intuitivo y adaptable a diferentes necesidades empresariales, permitiendo una fácil personalización. La arquitectura MVC utilizada en el desarrollo del sistema garantiza una clara separación entre la lógica de negocios, la presentación y el control, facilitando así el mantenimiento y la escalabilidad del sistema.

- **Abstract**

The main objective of the project is to develop an automated system that facilitates the vehicle rental process through an online platform. This system will allow users to register and make vehicle reservations, while administrators can manage vehicle inventory, bookings, and customers. Additionally, it will include report generation and email notification functionalities.

The system is designed to be intuitive and adaptable to various business needs, enabling easy customization. The MVC architecture used in the system development ensures a clear separation between business logic, presentation, and control, thereby facilitating system maintenance and scalability.



III. OBJETIVOS

1. Objetivo General

Desarrollar un sistema automatizado que optimice la gestión de alquiler de vehículos, mejorando la eficiencia operativa y la experiencia del cliente.

Objetivos Específicos

2. Objetivos Específicos

- Implementar la funcionalidad de registro y autenticación de usuarios: Garantizando la seguridad y privacidad de la información.
- Desarrollar un módulo para la gestión de inventarios de vehículos: Permitiendo agregar, modificar y eliminar vehículos.
- Facilitar el proceso de reserva y alquiler de vehículos: Asegurando que los usuarios puedan realizar estas acciones de manera sencilla y rápida.
- Proveer herramientas administrativas para la gestión eficiente de clientes y reservas: Incluyendo la posibilidad de modificar y cancelar reservas.
- Integrar un sistema de notificaciones por correo electrónico: Para enviar confirmaciones y recordatorios a los usuarios.
- Implementar la generación de reportes detallados: Para análisis y toma de decisiones estratégicas.
- Este proyecto busca

IV. PLANTEAMIENTO DEL PROBLEMA

1. Problema

Las empresas de alquiler de vehículos enfrentan diversos desafíos en la gestión eficiente de sus operaciones. Los sistemas manuales o no automatizados pueden llevar a errores en las reservas, duplicaciones y una experiencia de usuario insatisfactoria. Además, la falta de un sistema centralizado dificulta la gestión efectiva del inventario de vehículos, las reservas y los clientes.

Un sistema manual de gestión puede resultar en una operación ineficiente, pérdida de clientes y una mala reputación para la empresa. Por lo tanto, es necesario un sistema automatizado que permita gestionar estos procesos de manera eficiente, reduciendo los errores humanos y mejorando la satisfacción del cliente.



2. Justificación

La implementación de un sistema automatizado para la gestión de alquiler de vehículos es crucial para aumentar la eficiencia operativa y reducir los errores. Un sistema centralizado proporciona una plataforma para gestionar de manera efectiva las reservas, los clientes y el inventario de vehículos, mejorando así la experiencia del usuario y la operatividad del negocio.

Un sistema automatizado permite realizar tareas de manera más rápida y precisa, reduciendo la carga de trabajo manual y permitiendo a los empleados concentrarse en tareas más estratégicas. Además, mejora la satisfacción del cliente al ofrecer un proceso de reserva sencillo y confiable, y proporciona una mejor visibilidad y control sobre las operaciones del negocio.

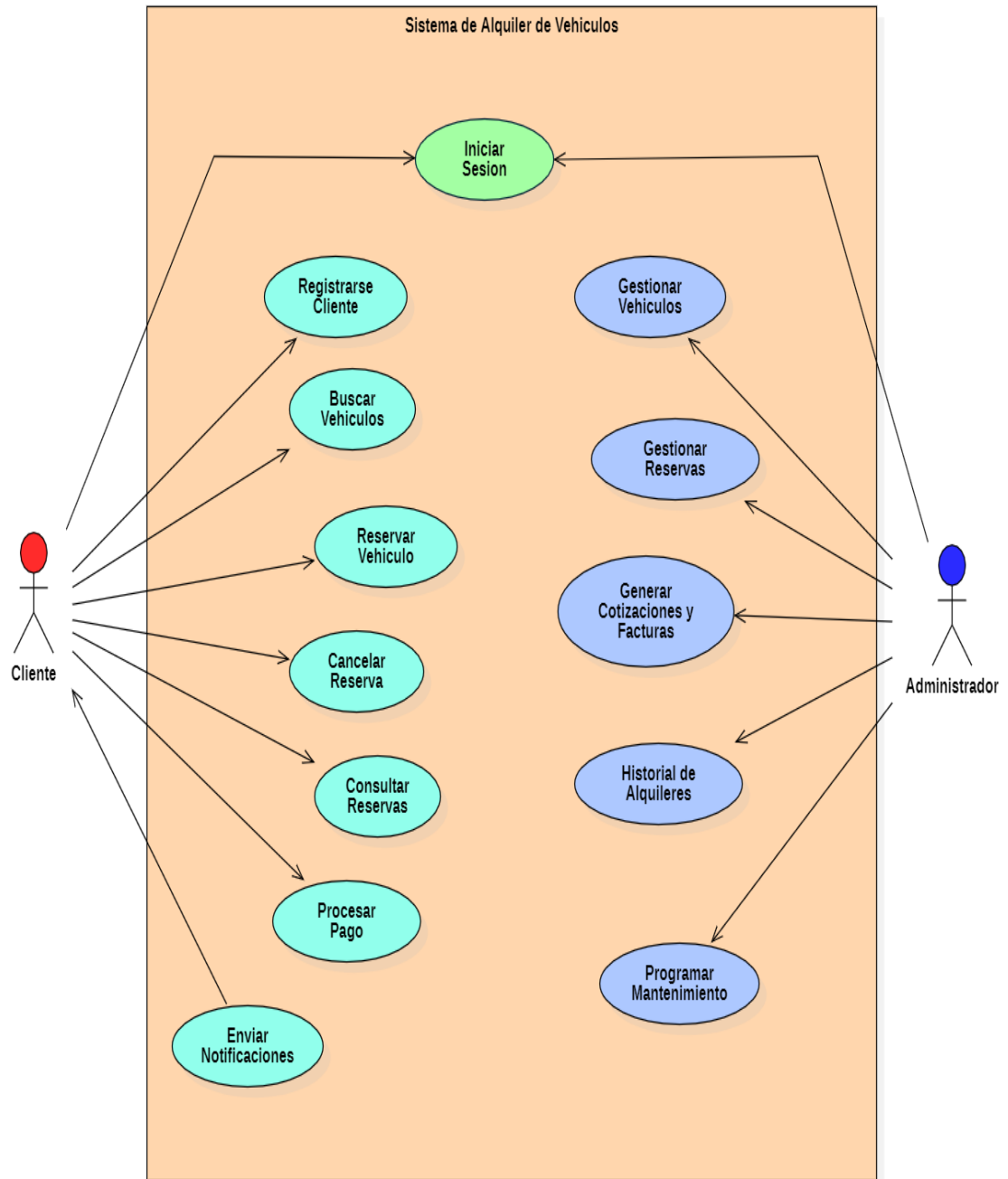
3. Alcance

El sistema abarcará:

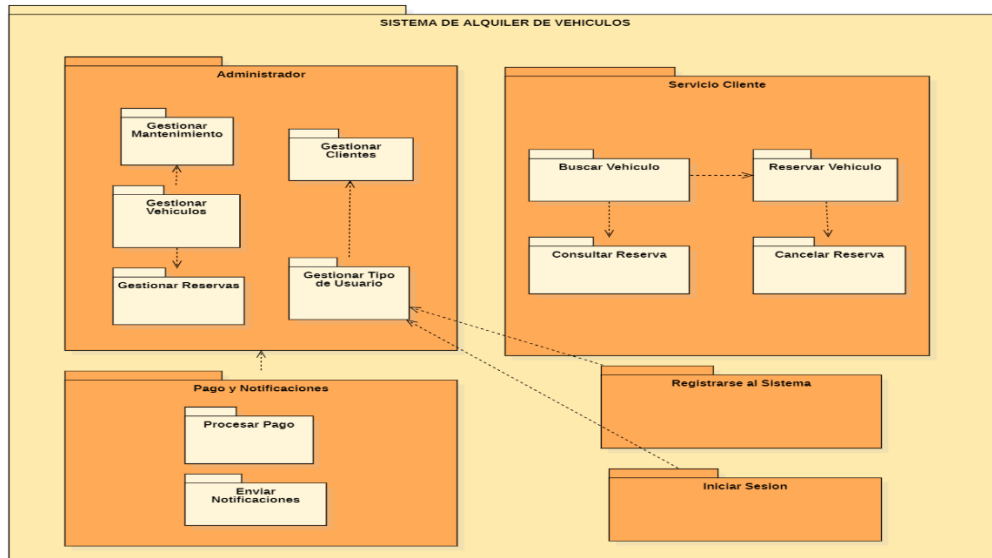
- Gestión de inventarios de vehículos: Permitiendo la alta, baja y modificación de vehículos en el sistema.
- Procesos de reserva y alquiler de vehículos: Los usuarios podrán realizar reservas y alquilar vehículos de manera sencilla y segura.
- Administración de clientes: Gestión de información de los clientes y sus reservas.
- Generación de reportes: Reportes detallados sobre las actividades del sistema para análisis y toma de decisiones estratégicas.
- Envío de notificaciones por correo electrónico: Confirmaciones y recordatorios automáticos para los usuarios.
- Interfaz de usuario intuitiva y personalizable: Adaptable a diferentes tipos de negocios de alquiler de vehículos.

V. DESARROLLO DEL TRABAJO

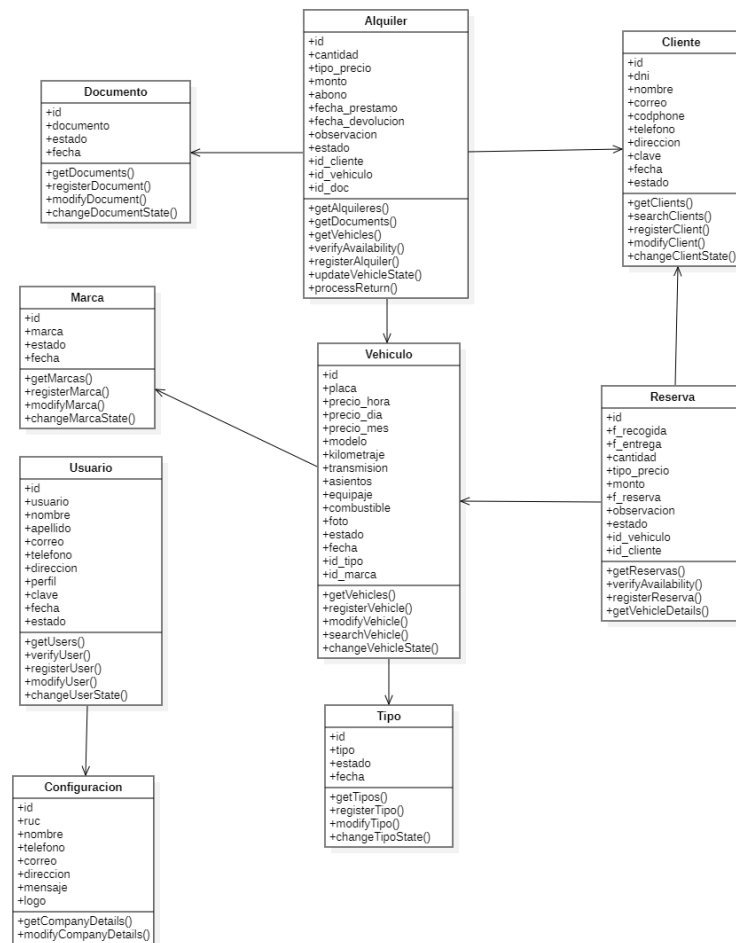
1. Diagrama de Caso de Uso



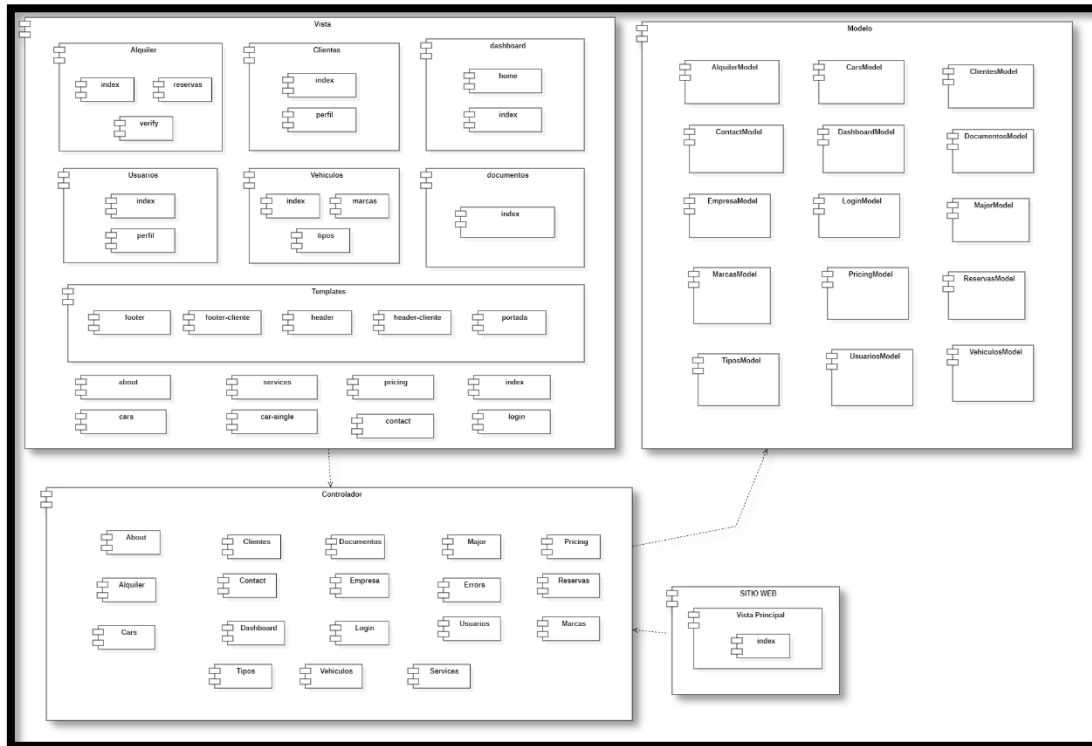
2. Diagrama de Paquetes



3. Diagrama de Clases



4. Crear Diagrama de Componentes



5. Analisis de la aplicación con SonarQube y Snyk

- SonarQube

Paso 01 : Se esta creando un repositorio público en tu cuenta personal llamado "SisAlquilerDeVehiculos", el cual contiene todos los archivos de tu proyecto y su historial de revisiones. Puedes inicializarlo con un archivo README y un archivo .gitignore, y elegir una licencia para tu código.

Owner * BCZLopezCatunta / Repository name * SisAlquilerDeVehiculos

Great repository names are short and memorable. Need inspiration? How about [symmetrical-adventure](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)



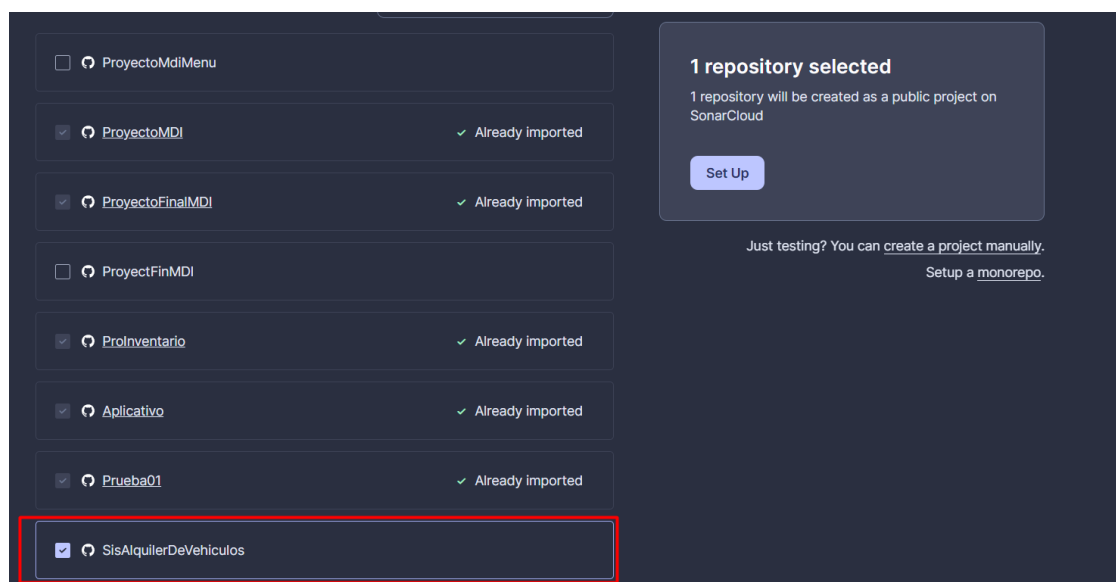
Paso 02: Se ha creado un nuevo repositorio en GitHub con la dirección URL "https://github.com/BCZLopezCatunta/SisAlquilerDeVehiculos.git" y ha cambiado la rama local existente a "main". Luego, ha agregado este repositorio a su sistema local en la carpeta "SisAlquilerDeVehiculos" en el escritorio y ha establecido una conexión remota con el repositorio de GitHub.

```
MINGW64:/c/Users/ZeroG/Desktop/SisAlquilerDeVehiculos
create mode 100644 Views/login.php
create mode 100644 Views/pricing.php
create mode 100644 Views/services.php
create mode 100644 Views/usuarios/index.php
create mode 100644 Views/usuarios/perfil.php
create mode 100644 Views/vehiculos/index.php
create mode 100644 Views/vehiculos/marcas.php
create mode 100644 Views/vehiculos/tipos.php
create mode 100644 index.php

ZeroG@DESKTOP-3D281KM MINGW64 ~/Desktop/SisAlquilerDeVehiculos (main)
$ git push origin main
Enumerating objects: 1603, done.
Counting objects: 100% (1603/1603), done.
Delta compression using up to 12 threads
Compressing objects: 100% (1576/1576), done.
Writing objects: 100% (1602/1602), 12.51 MiB | 1.97 MiB/s, done.
Total 1602 (delta 314), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (314/314), done.
To https://github.com/BCZLopezCatunta/SisAlquilerDeVehiculos.git
0f6264a..ba05e91 main -> main

ZeroG@DESKTOP-3D281KM MINGW64 ~/Desktop/SisAlquilerDeVehiculos (main)
$
```

Paso 03: Seleccionamos el repositorio "SisAlquilerDeVehiculos" de la organización BCZLopezCatunta en GitHub para configurarlo en SonarCloud. Ahora, después de seleccionar el repositorio deseado, das clic en la opción "Setup" para continuar con la configuración y prepararlo para su análisis en SonarCloud. Este proceso te permitirá evaluar la calidad del código de tu proyecto "SisAlquilerDeVehiculos" utilizando las herramientas y funcionalidades de SonarCloud.



Paso 04: Seleccionamos "Previous Version" en el contexto dado, estás especificando que la nueva codificación para el proyecto se basará en la versión anterior del código. Esto significa que cualquier parte del código que haya sido modificada desde la última versión se considerará como "nuevo código". Esta opción es recomendada para proyectos que siguen versiones o lanzamientos regulares. Una vez que has seleccionado esta opción, al hacer clic en "Create project".

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code.

This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology.

Learn more: [New Code Definition](#)

Set a new code definition for your organisation to use it by default for all new projects

This can help you use the Clean as You Code methodology consistently across projects.

[BCZLopezCatunta - Administration - New Code](#)

The new code for this project will be based on:

☒ **Previous version**

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

☐ **Number of days**

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.

Recommended for projects following continuous delivery.

☐ You can change this at any time in the project administration

[Back](#) [Create project](#)

Paso 05: El Análisis de código realizado por SonarSource en el proyecto "SisAlquilerDeVehiculos" de BCZLopezCatunta muestra que el proyecto tiene la calidad del código se evalúa en un 5,2% Duplications, lo que sugiere que hay espacio para mejorar. En cuanto a la seguridad, se han detectado 429 bugs y 4 vulnerabilidades.





Paso 06: Se pasa a resolver las vulnerabilidades

- En el archivo index.php del porque el problema:

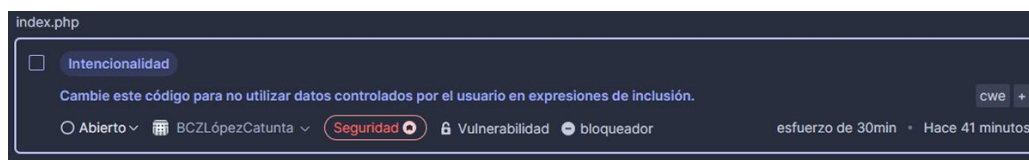
Impacto potencial de las inyecciones de inclusión

Si una aplicación es vulnerable a inyecciones de inclusión, puede ser susceptible a ataques dirigidos al servidor subyacente. Un usuario malintencionado puede crear solicitudes que filtren datos valiosos o logren la ejecución remota de código en el sitio web del servidor. El atacante puede llevar a cabo estos ataques sin depender de requisitos previos. El impacto de la vulnerabilidad depende de las medidas de control de acceso adoptadas en el sistema operativo de destino. En el peor de los casos, el proceso se ejecuta con privilegios de root, lo que significa que cualquier comando o programa del sistema operativo puede verse afectado.

Escenarios del mundo real que ilustran el impacto

a. Denegación de servicio y fuga de datos: El ataque tiene como objetivo interrumpir las actividades de la organización y beneficiarse de la filtración de datos. El atacante puede descargar datos del servidor interno para venderlos, modificar datos, enviar malware, detener servicios o agotar recursos. Esta amenaza es especialmente peligrosa si la organización atacada no tiene un plan de recuperación ante desastres.

b. Escalada y pivote de privilegios de root En este caso, el atacante puede elevar sus privilegios a nivel administrativo y atacar otros servidores. El impacto dependerá de la concentración de la empresa objetivo en su Defensa en Profundidad.



- En el archivo principal.php del porque el problema:

Impacto potencial de la falta de contraseña en una base de datos

Cuando una base de datos no requiere una contraseña para la autenticación, se abre la posibilidad de acceso y manipulación de los datos almacenados por parte de cualquier persona. Explotar esta vulnerabilidad puede tener varios impactos potenciales:

a. Acceso no autorizado a datos confidenciales: La ausencia de una contraseña para la autenticación permite que personas no autorizadas

obtengan acceso a datos confidenciales, como información de identificación personal, registros financieros y propiedad intelectual. Esto puede provocar robo de identidad, pérdidas financieras y daños a la reputación.

b. Compromiso de la integridad del sistema: Personas no autorizadas pueden obtener acceso ilimitado a la base de datos, lo que compromete la integridad del sistema. Los atacantes pueden inyectar código malicioso, alterar configuraciones o manipular datos, lo que puede provocar fallos en el funcionamiento del sistema y exposición a mayores riesgos de seguridad.

c. Modificaciones o eliminaciones no deseadas: La falta de una contraseña para el acceso a la base de datos permite que cualquier persona realice modificaciones o eliminaciones de los datos almacenados. Esto plantea un riesgo significativo, ya que los cambios no autorizados pueden provocar daños en los datos, pérdida de información crítica o la introducción de contenido malicioso.



- En el archivo categoria_producto.php del porque el problema:

Impacto Potencial de las Inyecciones de Bases de Datos

Las inyecciones de bases de datos, como las inyecciones SQL, representan una seria amenaza para la seguridad de las aplicaciones web. Cuando una aplicación es vulnerable a este tipo de ataques, los atacantes pueden manipular consultas de base de datos para ejecutar comandos maliciosos y comprometer la integridad de los datos. Esta vulnerabilidad expone a la aplicación a una serie de riesgos, que pueden tener impactos significativos en la seguridad y el funcionamiento de las bases de datos afectadas.

Suplantación de Identidad y Manipulación de Datos

La explotación de una consulta maliciosa puede conducir a una escalada de privilegios o a la fuga de datos sensibles almacenados en las bases de datos. Esta amenaza representa uno de los impactos más extendidos, ya que los atacantes pueden obtener acceso no autorizado a información confidencial, lo que a su vez puede resultar en la suplantación de identidad y manipulación de datos.

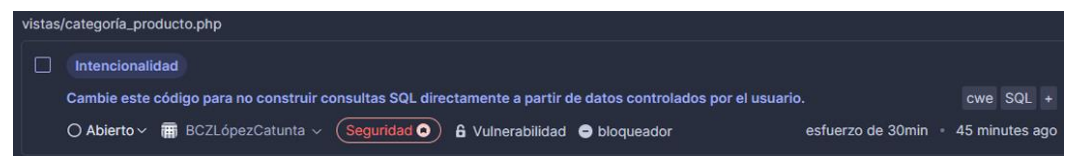
Eliminación de Datos y Denegación de Servicio

Los atacantes pueden utilizar consultas maliciosas para eliminar datos de las bases de datos afectadas, lo que puede tener consecuencias

devastadoras si la organización no cuenta con un plan de recuperación ante desastres. Esta amenaza es particularmente insidiosa, ya que puede resultar en la pérdida irreversible de información crítica y en la interrupción del servicio.

Encadenamiento de Inyecciones de Bases de Datos con Otras Vulnerabilidades

Los atacantes que explotan las inyecciones SQL suelen buscar maximizar sus ganancias mediante el aprovechamiento de otras vulnerabilidades. Esto puede incluir la filtración de secretos almacenados sin cifrar en las bases de datos, comprometiendo otros componentes del sistema, o incluso la ejecución remota de código si el sistema operativo del servidor o los permisos de la base de datos están mal configurados.



- En el archivo product_img.php del porque el problema:

Las inyecciones de bases de datos, como las inyecciones SQL, representan una seria amenaza para la seguridad de las aplicaciones web. Cuando una aplicación es vulnerable a este tipo de ataques, los atacantes pueden manipular consultas de base de datos para ejecutar comandos maliciosos y comprometer la integridad de los datos. Esta vulnerabilidad expone a la aplicación a una serie de riesgos, que pueden tener impactos significativos en la seguridad y el funcionamiento de las bases de datos afectadas.

Suplantación de Identidad y Manipulación de Datos

La explotación de una consulta maliciosa puede conducir a una escalada de privilegios o a la fuga de datos sensibles almacenados en las bases de datos. Esta amenaza representa uno de los impactos más extendidos, ya que los atacantes pueden obtener acceso no autorizado a información confidencial, lo que a su vez puede resultar en la suplantación de identidad y manipulación de datos.

Eliminación de Datos y Denegación de Servicio

Los atacantes pueden utilizar consultas maliciosas para eliminar datos de las bases de datos afectadas, lo que puede tener consecuencias devastadoras si la organización no cuenta con un plan de recuperación ante desastres. Esta amenaza es particularmente insidiosa, ya que puede resultar en la pérdida irreversible de información crítica y en la interrupción del servicio.

Encadenamiento de Inyecciones de Bases de Datos con Otras Vulnerabilidades

Los atacantes que explotan las inyecciones SQL suelen buscar maximizar sus ganancias mediante el aprovechamiento de otras vulnerabilidades. Esto puede incluir la filtración de secretos almacenados sin cifrar en las bases de datos, comprometiendo otros componentes del sistema, o incluso la ejecución remota de código si el sistema operativo del servidor o los permisos de la base de datos están mal configurados.

- Método y Técnicas Usadas en el SonarQube
 - o SonarSource

a)El problema index.php:

```
//index.php
5  b12820...
6      <?phpinclude "../inc/head.php"; ?>
7      </cabeza>
8      <?php
9
10         si(!Está establecido($_GET['vista']) || $_GET['vista']==""){
11             $_GET['vista']="acceso";
12         }
13
14
15         si(es_archivo("../vistas/".$_GET['vista'].".php") && $_GET['vista']!="acceso" && $_GET['vista']!="404"){
16
17             /*= Cerrar sesión =*/
18             si(!Está establecido($_SESSION['identificación']) || $_SESSION['identificación']=="") || (!Está establecido
19             ($_SESSION['usuario']) || $_SESSION['usuario']==""){
20                 include "../vistas/logout.php";
21                 salida();
22             }
23
24             incluir "../inc/navbar.php";
25
26             4 incluir 3 "../vistas/".$_GET['vista'].".php";
27
28             incluir "../inc/script.php";
29
30             }de más{
31                 si($_GET['vista']=="acceso"){
32                     incluir "../vistas/login.php";
33                 }de más{
34                     incluir "../vistas/404.php";
35                 }
36             }
37         }
38     }
39 }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
25
```

Es importante destacar que esta validación debe realizarse en el lado del servidor para garantizar su efectividad. Al implementar esta técnica, se reduce significativamente el riesgo de ataques de inyección en las expresiones de inclusión, lo que contribuye a fortalecer la seguridad de la aplicación.

Solución del Problema:

```
index.php
1 <?php
2 define('VISTAS_PATH', "../vistas/");
3
4 $INCLUDE_ALLOW_LIST = [
5     'casa.php',
6     'tablero.php',
7     'perfil.php',
8     'configuración.php'
9 ];
10
11 // Función para verificar la sesión activa
12 function verificarSesionActiva() {
13     if (empty($_SESSION['identificación']) || empty($_SESSION['us
14         require_once VISTAS_PATH . 'cerrar sesión.php';
15         exit();
16     }
17 }
18
19 // Obtener la vista solicitada
20 $vista = $_GET['vista'] ?? '';
21
22 // Verificar si la vista solicitada está en la lista permitida y
23 if ($vista && in_array($vista, $INCLUDE_ALLOW_LIST) && is_file(VI
24     require_once VISTAS_PATH . $vista . ".php";
25 } else {
26     // Si la vista no está permitida o no existe, cargar la página
27     require_once VISTAS_PATH . '404.php';
28 }
29
30 // Iniciar sesión si la vista solicitada es 'acceso'; de lo contr
31 if ($vista === 'acceso') {
32     require_once VISTAS_PATH . 'iniciar sesión.php';
33 } else {
34     verificarSesionActiva();
35 }
36
37 // Incluir la barra de navegación
38 require_once "../inc/navbar.php";
39 ?>
```

Se utilizan las siguientes metodologías y técnicas:

- *Lista de Permitidos:* Se define una lista de archivos permitidos y seguros que la aplicación puede incluir. Esto se logra mediante la variable `$INCLUDE_ALLOW_LIST`, que contiene los nombres de los archivos autorizados, como `'casa.php'`, `'tablero.php'`, `'perfil.php'` y `'configuración.php'`.
- *Validación de Sesión Activa:* Se implementa una función llamada `verificarSesionActiva()` que verifica si la sesión del usuario está activa. Si la identificación del usuario o el usuario mismo están vacíos, se requiere el archivo `'cerrar sesión.php'` desde la ruta de vistas y se finaliza la ejecución del script.
- *Validación de Vista Solicitada:* Se verifica si la vista solicitada por el usuario está en la lista permitida y si existe como archivo. Esto se realiza mediante la variable `$vista`, que obtiene el valor del parámetro "vista" en la URL. Si la vista está en la lista permitida y es un archivo válido, se incluye el archivo correspondiente; de lo contrario, se carga la página `'404.php'`.
- *Inclusión Condicional de Archivos:* Dependiendo de la vista solicitada, se realizan inclusiones condicionales de archivos. Si la vista solicitada es `'acceso'`, se requiere el archivo `'iniciar sesión.php'`; de lo contrario, se verifica la sesión activa llamando a la función `verificarSesionActiva()`.
- *Inclusión de Barra de Navegación:* Finalmente, se incluye la barra de navegación desde el archivo `"../inc/navbar.php"`.

b) El problema main.php:

```
php/ principal.php Ver todos los problemas en este archivo

1  bl2020...  <?php
2
3      # Conexión a la base de datos #
4      función conexión(){
5          $dop=nuevoPDO('mysql:host=localhost;dbname=pdo','raiz','');
6
7          devolver $dop;
8      }
9
10     #Verificar datos#
11     función verificar_datos($filtro,$cadena){
12         si(ereg_match('/'. $filtro.'/', $cadena)){
13             devolver FALSO;
14         }deñás{
15             devolver verdadero;
16         }
17     }
18 }
```

El código presenta una vulnerabilidad al utilizar una contraseña vacía para conectarse a una base de datos MySQL. Para solucionar esta vulnerabilidad, es recomendable utilizar una contraseña segura recuperada de una variable de entorno, como `MYSQL_SECURE_PASSWORD`, que se establece durante la implementación y debe ser única y robusta para cada base de datos.

Es importante evitar el uso de contraseñas codificadas en el código, ya que esto puede exponer la seguridad de la aplicación a riesgos significativos. Las contraseñas codificadas pueden ser descubiertas fácilmente por desarrolladores o atacantes, lo que podría resultar en acceso no autorizado a la base de datos y posibles violaciones de datos. Además, el uso de contraseñas codificadas dificulta la flexibilidad para cambiar las contraseñas sin modificar el código, lo que puede generar problemas de control de versiones y representar un riesgo para la seguridad.

Para mitigar estos riesgos, se recomienda utilizar métodos seguros para almacenar y recuperar contraseñas, como el uso de variables de entorno, archivos de configuración o sistemas seguros de administración de claves. Estas prácticas permiten una mayor seguridad, flexibilidad y separación

de información confidencial del código base, lo que contribuye a la protección de los datos y a la prevención de posibles vulnerabilidades.

Solución del Problema:

```
main.php x
php > main.php
1 <?php
2
3 # Conexion a la base de datos #
4 function conexion(){
5     $db_host = 'localhost';
6     $db_name = 'pdo';
7     $db_user = 'root';
8     $db_pass = getenv('DB_PASSWORD'); // Obtener la contraseña desde una variable de entorno
9
10    try {
11        $pdo = new PDO("mysql:host=$db_host;dbname=$db_name", $db_user, $db_pass);
12        return $pdo;
13    } catch (PDOException $e) {
14        die("Error de conexión: " . $e->getMessage());
15    }
16 }
17
18 # Verificar datos #
19 function verificar_datos($filtro,$cadena){
20     if(preg_match("/".$filtro."/", $cadena)){
21         return false;
22     }else{
23         return true;
24     }
25 }
26 }
```

Se utilizan las siguientes metodologías y técnicas:

- **Uso de Variables de Entorno:** Se utiliza la función `getenv()` para recuperar la contraseña de la base de datos desde una variable de entorno llamada 'DB_PASSWORD'. Esto asegura que la contraseña no esté codificada en el código y se obtenga de una fuente externa, lo que mejora la seguridad.
- **Conexión Segura a la Base de Datos:** Al utilizar la contraseña recuperada de la variable de entorno para establecer la conexión a la base de datos, se garantiza que se esté utilizando una contraseña segura al conectarse a la base de datos MySQL. Esto es fundamental para proteger la integridad de los datos y prevenir posibles violaciones de seguridad.
- **Manejo de Excepciones:** Se utiliza un bloque `try-catch` para manejar posibles errores de conexión a la base de datos. Esto asegura que cualquier error sea capturado y manejado adecuadamente, lo que contribuye a la robustez y estabilidad del sistema.

c) El problema categoria_producto.php:

```
24 $categorias=null;
25
26
27 </div>
28 <div class="column">
29 <?php
30     $categoria_id = (isset($_GET['categoria_id'])) ? $_GET['categoria_id'] : 0;
31
32     /*= Verificando categoria =*/
33     $check_categoria=conexion();
34     $check_categoria->consulta( "SELECT * FROM categoria WHERE categoria_id= $categoria_id");
35
36     $categoria_id = $check_categoria->fetch(PDO::FETCH_ASSOC);
37
38     if($check_categoria->ContarFilas()>0){
39
40         $check_categoria->buscar();
41
42         echo '
43         <h2 class="titulo tiene-texto-centrado">'. $check_categoria['categoria_nombre']. '</h2>
44         <p class="ha-texto-centrado pb-6">'. $check_categoria['categoria_ubicacion']. '</p>
45     ';
```

El código proporcionado presenta una vulnerabilidad de inyección de SQL, ya que los datos controlados por el usuario se insertan directamente en una cadena de consulta, lo que puede permitir a un atacante realizar consultas maliciosas. Para solucionar esta vulnerabilidad, se recomienda utilizar declaraciones preparadas en lugar de construir consultas SQL directamente a partir de datos controlados por el usuario. La solución compatible implica el uso de declaraciones preparadas, donde los valores de los parámetros se pasan por separado, lo que evita la posibilidad de inyección de SQL. Al utilizar declaraciones preparadas, la lógica de la consulta se compila antes de que se pasen los valores de los parámetros, lo que evita que los datos controlados por el usuario afecten la consulta subyacente.

Solución del Problema:

```
24 </div>
25
26 <div class="column">
27 <?php
28     $categoria_id = (isset($_GET['category_id'])) ? $_GET['category_id'] : 0;
29
30     /*= Verificando categoria =*/
31     $pdo = conexion();
32     $statement = $pdo->prepare("SELECT * FROM categoria WHERE categoria_id = :categoria_id");
33     $statement->bindParam(':categoria_id', $categoria_id, PDO::PARAM_INT);
34     $statement->execute();
35     $result = $statement->fetch(PDO::FETCH_ASSOC);
```

Se utilizan las siguientes metodologías y técnicas:

- Se utiliza la técnica de declaración preparada para protegerse contra la inyección de SQL. En lugar de construir la consulta SQL directamente a partir de datos controlados por el usuario, se utilizan marcadores de posición en la consulta y luego se enlazan los valores de los parámetros a estos marcadores de posición. Esto evita que los datos controlados por el usuario afecten la lógica de la consulta y previene posibles ataques de inyección de SQL.

Además, el código utiliza la función `isset()` para verificar si el parámetro 'category_id' está presente en la URL antes de asignarlo a la variable `$categoria_id`. Esto ayuda a evitar errores de ejecución si el parámetro no está presente en la URL.

d) El problema product_img.php:

```
1  <?php
2  include_once "../inc/btn_back.php";
3  require_once "../php/main.php";
4
5  $id = (isset($_GET['product_id_up'])) ? $_GET['product_id_up'] : 0;
6  /*== Verificando producto ==*/
7  $pdo = conexion();
8  $statement = $pdo->prepare("SELECT * FROM producto WHERE producto_id = :id");
9  $statement->bindParam(':id', $id, PDO::PARAM_INT);
10 $statement->execute();
11 $result = $statement->fetch(PDO::FETCH_ASSOC);
12
13 if($check_producto->rowCount()>0){
14     $datos=$check_producto->fetch();
15 }
16
17 <div class="form-rest mb-6 mt-6"></div>
18 <div class="column">
19 <div class="column es-dos-quintos">
```

El código proporcionado presenta una vulnerabilidad de inyección de SQL, ya que los datos controlados por el usuario se insertan directamente en una cadena de consulta, lo que puede permitir a un atacante realizar consultas maliciosas. Para solucionar esta vulnerabilidad, se recomienda utilizar declaraciones preparadas en lugar de construir consultas SQL directamente a partir de datos controlados por el usuario.

La solución compatible implica el uso de declaraciones preparadas, donde los valores de los parámetros se pasan por separado, lo que evita la posibilidad de inyección de SQL. Al utilizar declaraciones preparadas, la lógica de la consulta se compila antes de que se pasen los valores de los parámetros, lo que evita que los datos controlados por el usuario afecten la consulta subyacente.

Solución del Problema:

```
<?php
include_once "../inc/btn_back.php";
require_once "../php/main.php";

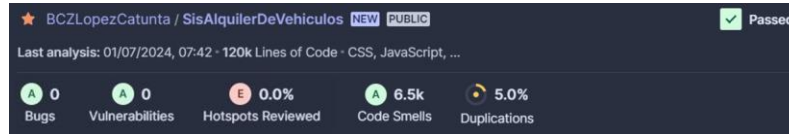
$id = (isset($_GET['product_id_up'])) ? $_GET['product_id_up'] : 0;
/*== Verificando producto ==*/
$pdo = conexion();
$stmt = $pdo->prepare("SELECT * FROM producto WHERE producto_id = :id");
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_ASSOC);

if($check_producto->rowCount()>0){
    $datos=$check_producto->fetch();
}
```

Se utilizan las siguientes metodologías y técnicas:

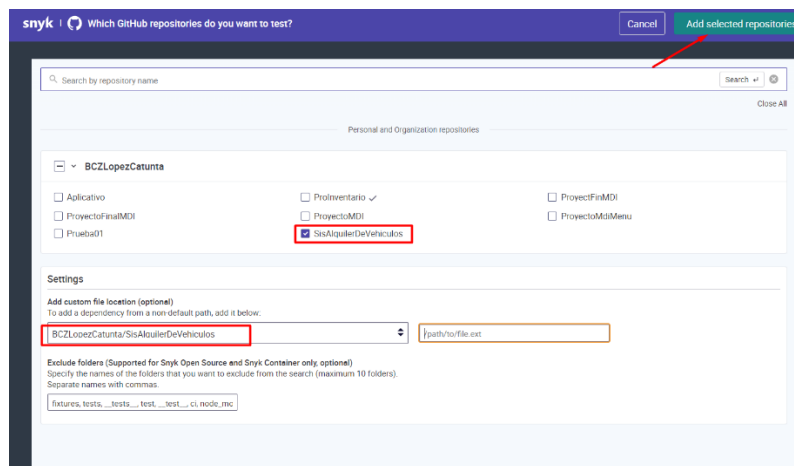
El código proporcionado utiliza la técnica de declaración preparada para protegerse contra la inyección de SQL. En lugar de construir la consulta SQL directamente a partir de datos controlados por el usuario, se utilizan marcadores de posición en la consulta y luego se enlazan los valores de los parámetros a estos marcadores de posición. Esto evita que los datos controlados por el usuario afecten la lógica de la consulta y previene posibles ataques de inyección de SQL. Además, el código utiliza la función `isset()` para verificar si el parámetro `'product_id_up'` está presente en la URL antes de asignarlo a la variable `$id`. Esto ayuda a evitar errores de ejecución si el parámetro no está presente en la URL.

- Solución del Problema con el SonarQube
 - Existencia de bugs: 0
 - ● Vulnerabilidades: 0.0%
 - ● Hotspots revisados: 0.0%
 - ● Código con olor a bug: 0
 - Código Smells: 6.5k
 - ● Duplicaciones: 5.0%
 -

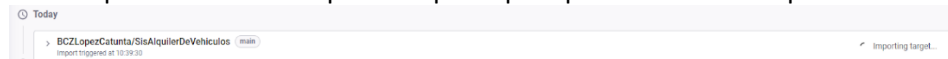


- Snky

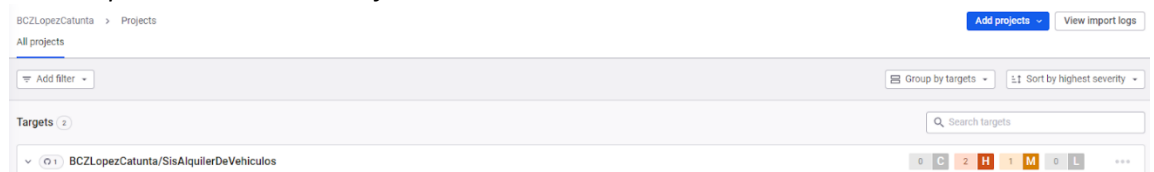
Para subir un proyecto a la plataforma de seguridad de código abierto Snky. Primero, el usuario busca el repositorio "SisAlquilerDeVehiculos" dentro de su cuenta de GitHub. Luego, selecciona el repositorio y hace clic en "Agregar repositorios seleccionados" para iniciar el proceso de análisis de seguridad.



En este paso observamos que se importa para poder analizar el Aplicativo subido:



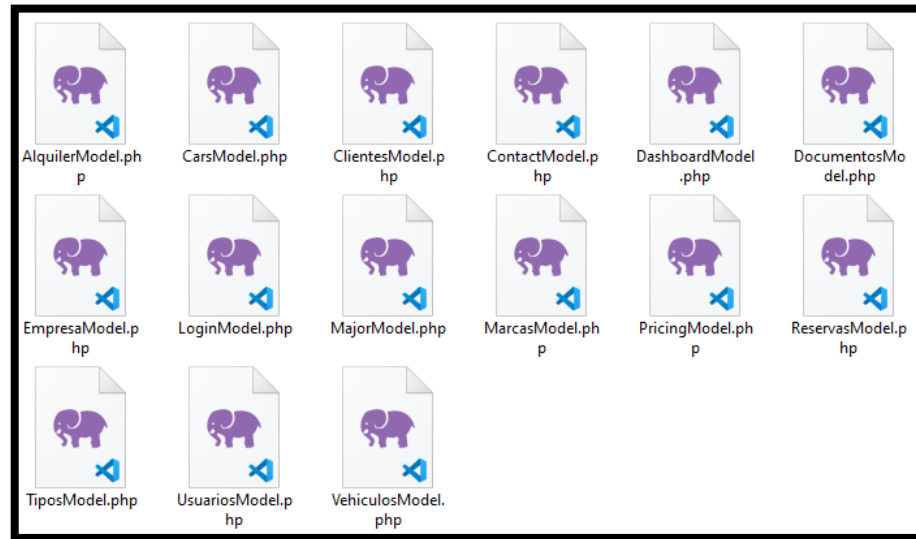
Se han realizado pruebas y se han detectado algunos errores de alta gravedad, incluyendo un error crítico (C), dos errores de alta prioridad (H), y un error de prioridad media (M). También se han encontrado algunos errores de baja prioridad (L) y algunos errores que aún no se han clasificado.



6. Reporte de Cobertura de Pruebas

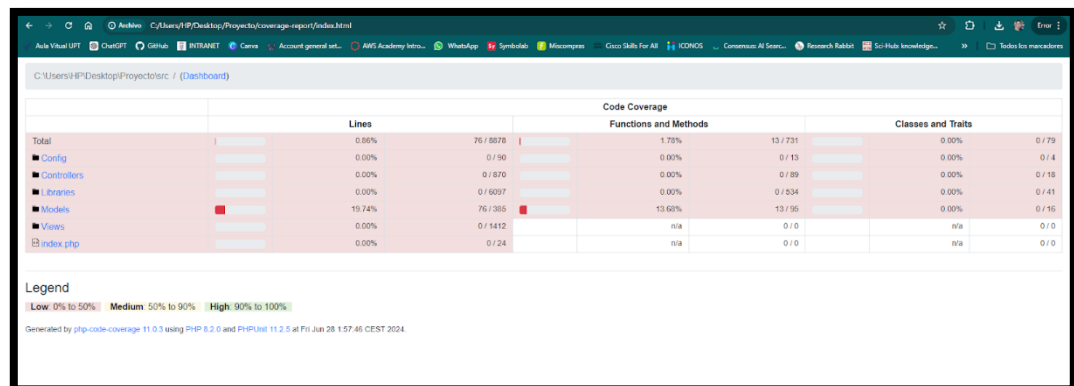
- Métodos para hacer las pruebas Tests "Models"

Se sugiere usar PHPUnit y Composer para realizar pruebas unitarias de estos modelos. Se puede utilizar Composer para instalar PHPUnit y las dependencias necesarias.



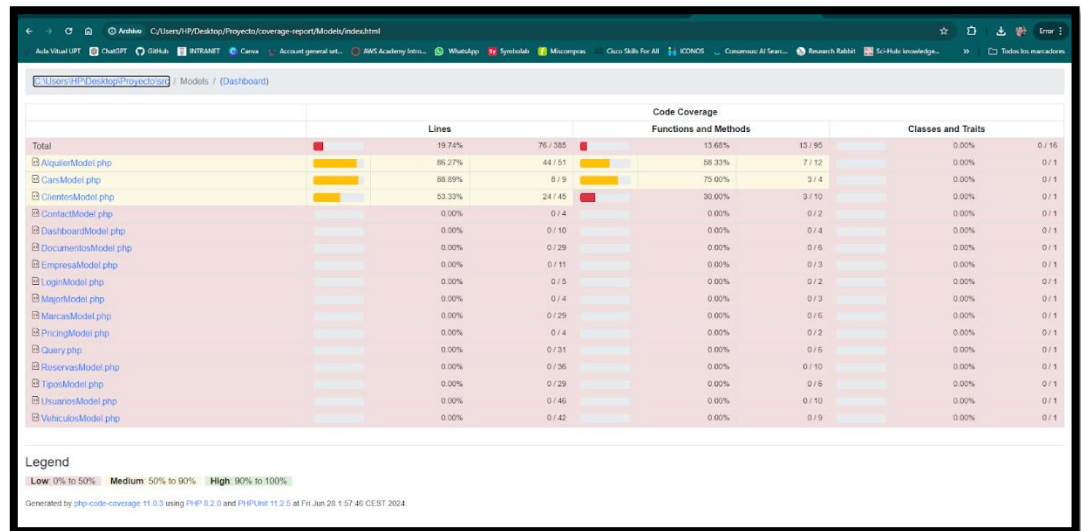
- Fase de Inicio para las Pruebas

El reporte está dividido en tres secciones: "Líneas", "Funciones y Métodos" y "Clases y Rasgos". Cada sección muestra el porcentaje de cobertura de código para cada elemento de la aplicación, incluyendo el número total de líneas, funciones, métodos y clases. La captura también incluye una leyenda que indica qué colores representan cada rango de cobertura. En general, el reporte indica que la aplicación tiene una baja cobertura de código, con la mayoría de los elementos teniendo una cobertura del 0%. Sin embargo, la sección "Modelos" tiene una cobertura del 19.74%, lo que indica que esta sección de la aplicación ha sido probada más a fondo que las demás.



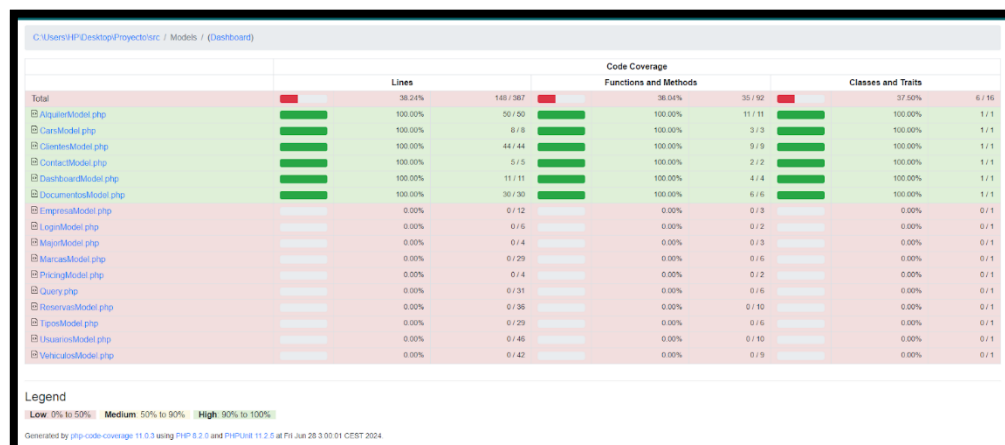
- **En este apartado podemos observar todos los metodos**

En la captura se muestra un informe de cobertura de código generado por la herramienta php-code-coverage. El informe muestra el porcentaje de líneas de código, funciones y métodos que están cubiertos por pruebas unitarias. El informe se divide en diferentes secciones, como "Total", "Lines", "Functions and Methods", y "Classes and Traits". Cada sección muestra el nombre del archivo PHP, el porcentaje de cobertura y el número de líneas de código, funciones y métodos que están cubiertos. El informe también muestra una leyenda con los colores que representan los diferentes niveles de cobertura.



- **En este apartado vemos el progreso de los tests**

Se puede observar que hay un total de 16 archivos PHP, de los cuales 8 tienen una cobertura de código del 100%, lo que significa que están totalmente cubiertos por pruebas unitarias. Estos archivos son: AlquilerModel.php, CarsModel.php, ClientesModel.php, ContactModel.php, DashboardModel.php, DocumentosModel.php, EmpresaModel.php y LoginModel.php. Los 8 archivos restantes tienen una cobertura de código del 0%.





- **En este otro Punto vemos los siguientes progresos**

La imagen muestra un informe de cobertura de código generado por la herramienta php-code-coverage. El informe muestra la cobertura de código para cada archivo PHP en un proyecto, desglosado por líneas, funciones y métodos, y clases y rasgos. La cobertura de código se muestra como un porcentaje, y el color del indicador de progreso refleja la cobertura: verde para 100%, amarillo para 50% a 90% y rojo para 0% a 50%. El informe también muestra la cantidad de líneas, funciones y métodos, y clases y rasgos que se han probado y la cantidad total.



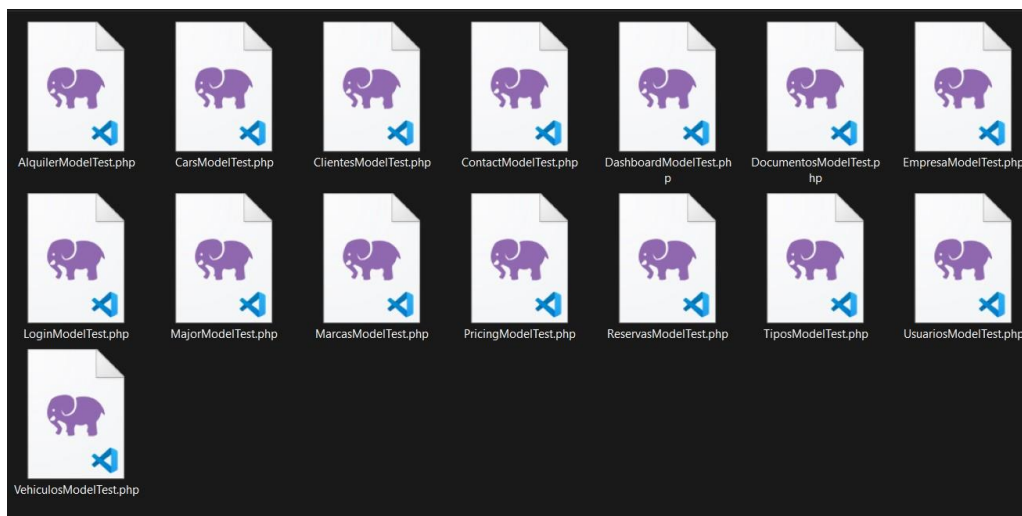
- **En este otro Punto como parte final podemos los testeos finales de los métodos**

El informe de cobertura de código muestra que la mayoría de los archivos PHP tienen una cobertura del 100%, excepto algunos que tienen una cobertura del 91.69% en líneas, 93.41% en funciones y métodos, y 93.75% en clases y rasgos, lo que indica una alta calidad en la prueba del código.



- **Pruebas Unitarias**

Se esta utilizando las prácticas de desarrollo de software sólidas, incluyendo pruebas unitarias, con el fin de asegurar la calidad del código de la aplicación de lo cual observamos en estos siguientes archivos.



- **Herramientas Necesarias para los Test**

Herramienta	Descripción	Requisitos y Pasos a usar
PHPUnit	PHPUnit es un marco de pruebas unitarias para PHP. Se utiliza para escribir y ejecutar pruebas automatizadas que verifican la funcionalidad del código.	1. Instalar PHPUnit: composer require --dev phpunit/phpunit 2. Crear archivos de prueba en la carpeta tests 3. Ejecutar pruebas: vendor/bin/phpunit
Composer	Composer es un gestor de dependencias para PHP. Administra las bibliotecas que tu proyecto necesita y facilita su instalación y actualización.	1. Instalar Composer desde getcomposer.org 2. Crear un archivo composer.json en el proyecto 3. Añadir PHPUnit como dependencia en composer.json 4. Ejecutar composer install para instalar las dependencias



Dentro del proyecto:

Elemento	Descripción
'composer.json'	Archivo para gestionar dependencias del proyecto, asegurando la inclusión de PHPUnit.
'phpunit.xml'	Configuración de PHPUnit para definir cómo y dónde ejecutar las pruebas.
'tests/Unit/'	Directorio donde se alojan todas las pruebas unitarias del proyecto.
'tests/Unit/ModelTest.php'	Ejemplo de archivo de prueba unitario para un modelo específico, como UserTest.php.
'vendor/'	Carpeta generada por Composer que contiene todas las dependencias instaladas, incluido PHPUnit.
'vendor/bin/phpunit'	Comando para ejecutar PHPUnit y realizar las pruebas unitarias definidas en tests/Unit/.

○ Archivos para el Test:

- Test de AlquilerModel.php

Título del Test	Descripción
testGetAlquiler	Verifica que el método getAlquiler devuelve los datos correctos cuando se llama al método selectAll.
<p>Código:</p> <pre>public function testGetAlquiler() { // Preparar los datos esperados \$mockedData = [['id' => 1, 'nombre' => 'Cliente A', 'placa' => 'ABC123', 'modelo' => 'Modelo X', 'documento' => 'Documento A', 'tipo' => 'Tipo A'], ['id' => 2, 'nombre' => 'Cliente B', 'placa' => 'XYZ789', 'modelo' => 'Modelo Y', 'documento' => 'Documento B', 'tipo' => 'Tipo B'],]; // Configurar el comportamiento esperado del mock de Query para el método selectAll \$this->mockQuery->shouldReceive('selectAll') ->once() ->andReturn(\$mockedData); // Instanciar el modelo y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->getAlquiler(); // Asegurar que el resultado coincide con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	



testGetEmpresa	Verifica que el método getEmpresa devuelve los datos correctos cuando se llama al método select con una consulta específica.
<p>Codigo:</p> <pre>public function testGetEmpresa() { // Datos simulados que se esperan retornar del método select \$mockedData = [['id' => 1, 'nombre' => 'Empresa A', 'direccion' => 'Calle Principal', 'telefono' => '123456789'],]; // Configurar el mock para el método select en la consulta de empresa \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM configuracion") ->andReturn(\$mockedData); // Instanciar AlquilerModel y llamar al método getEmpresa \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->getEmpresa(); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRegistrarAlquilerSuccess	Verifica que el método registrarAlquiler registra un alquiler correctamente cuando no hay conflictos y la inserción es exitosa.
<p>Codigo:</p> <pre>public function testRegistrarAlquilerSuccess() { // Preparar los datos para el registro de alquiler \$cantidad = 1; \$precios = 'tipo'; \$monto = 5; \$abono = 15; \$fecha = '2022-05-10'; \$fecha_devolucion = '2022-05-10'; \$observacion = 'si'; \$id_cli = 1; \$id_veh = 1; \$documento = 5;</pre>	



```
// Configurar el comportamiento esperado del mock de Query para simular que no existe
un registro previo
$this->mockQuery->shouldReceive('select')
    ->once()
    ->andReturnNull();

// Configurar el comportamiento esperado del mock de Query para el método insertar
// Simulamos que el insertar devuelve 1 para simular una inserción exitosa
$this->mockQuery->shouldReceive('insertar')
    ->once()
    ->andReturn(1);

// Instanciar el modelo y llamar al método que queremos probar
$alquilerModel = new AlquilerModel($this->mockQuery);
$result = $alquilerModel->registrarAlquiler($cantidad, $precios, $monto, $abono,
$fecha, $fecha_devolucion, $observacion, $id_cli, $id_veh, $documento);

// Asegurar que el resultado coincide con el valor retornado por insertar (1 en este
caso)
$this->assertEquals(1, $result);
}
```

testRegistrarAlquilerError

Verifica que el método registrarAlquiler devuelve "error" cuando la inserción falla.

Codigo:

```
public function testRegistrarAlquilerError()
{
    // Preparar los datos para el registro de alquiler
    $cantidad = 1;
    $precios = 'tipo';
    $monto = 5;
    $abono = 15;
    $fecha = '2022-05-10';
    $fecha_devolucion = '2022-05-10';
    $observacion = 'si';
    $id_cli = 1;
    $id_veh = 1;
    $documento = 5;

    // Configurar el comportamiento esperado del mock de Query para simular que no existe
un registro previo
    $this->mockQuery->shouldReceive('select')
```



```
->once()
->andReturnNull();

// Configurar el comportamiento esperado del mock de Query para el método insertar
// Simulamos que el insertar devuelve 0 para simular un fallo en la inserción
$this->mockQuery->shouldReceive('insertar')
->once()
->andReturn(0);

// Instanciar el modelo y llamar al método que queremos probar
$alquilerModel = new AlquilerModel($this->mockQuery);
$result = $alquilerModel->registrarAlquiler($cantidad, $precios, $monto, $abono,
$fecha, $fecha_devolucion, $observacion, $id_cli, $id_veh, $documento);

// Asegurar que el resultado coincide con el string "error"
$this->assertEquals('error', $result);
}
```

testRegistrarAlquilerExistente

Verifica que el método registrarAlquiler devuelve "existe" cuando ya existe un alquiler activo para los mismos datos.

Codigo:

```
public function testRegistrarAlquilerExistente()
{
    // Datos simulados para el registro de alquiler
    $cantidad = 1;
    $precios = 'Tipo A';
    $monto = 100;
    $abono = 50;
    $fecha = '2024-06-28';
    $fecha_devolucion = '2024-06-30';
    $observacion = 'Alquiler de prueba';
    $id_cli = 1;
    $id_veh = 1;
    $documento = 1;

    // Configurar el mock para el método select en la verificación de existencia de
    alquiler
    $this->mockQuery->shouldReceive('select')
        ->with("SELECT * FROM alquiler WHERE id_cliente = $id_cli AND id_vehiculo =
$id_veh AND id_doc = $documento AND estado = 1")
        ->andReturn([[ 'id' => 1 ]]); // Alquiler existe
}
```



<pre>// Instanciar AlquilerModel y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->registrarAlquiler(\$cantidad, \$precios, \$monto, \$abono, \$fecha, \$fecha_devolucion, \$observacion, \$id_cli, \$id_veh, \$documento); // Verificar que se retorna el resultado esperado en caso de existencia de alquiler ('existe') \$this->assertEquals('existe', \$result); }</pre>	
testGetDoc	Verifica que el método getDoc devuelve los datos correctos cuando se llama al método selectAll.
<p>Codigo:</p> <pre>public function testGetDoc() { // Preparar los datos esperados \$mockedData = [['id' => 1, 'nombre' => 'Documento A', 'estado' => 1], ['id' => 2, 'nombre' => 'Documento B', 'estado' => 1],]; // Configurar el comportamiento esperado del mock de Query para el método selectAll \$this->mockQuery->shouldReceive('selectAll') ->once() ->andReturn(\$mockedData); // Instanciar el modelo y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->getDoc(); // Asegurar que el resultado coincide con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	
testGetVehiculos	Verifica que el método getVehiculos devuelve los datos correctos cuando se llama al método selectAll.
<p>Codigo:</p> <pre>public function testGetVehiculos() { // Preparar los datos esperados \$mockedData = [['id' => 1, 'placa' => 'ABC123', 'id_tipo' => 1, 'id_marca' => 1, 'estado' => 1],</pre>	



```
1],
    ['id' => 2, 'placa' => 'XYZ789', 'id_tipo' => 2, 'id_marca' => 2, 'estado' =>
];

// Configurar el comportamiento esperado del mock de Query para el método selectAll
$this->mockQuery->shouldReceive('selectAll')
    ->once()
    ->andReturn($mockedData);

// Instanciar el modelo y llamar al método que queremos probar
$alquilerModel = new AlquilerModel($this->mockQuery);
$result = $alquilerModel->getVehiculos();

// Asegurar que el resultado coincide con los datos esperados
$this->assertEquals($mockedData, $result);
}
```

testGetVehiculo

Verifica que el método getVehiculo devuelve los datos correctos cuando se llama al método select con el ID del vehículo.

Codigo:

```
public function testGetVehiculo()
{
    // Preparar los datos esperados
    $mockedData = ['id' => 1, 'placa' => 'ABC123', 'id_tipo' => 1, 'id_marca' => 1,
'estado' => 1];

    // Configurar el comportamiento esperado del mock de Query para el método select
    $this->mockQuery->shouldReceive('select')
        ->once()
        ->andReturn($mockedData);

    // ID del vehículo a consultar
    $vehiculoId = 1;

    // Instanciar el modelo y llamar al método que queremos probar
    $alquilerModel = new AlquilerModel($this->mockQuery);
    $result = $alquilerModel->getVehiculo($vehiculoId);

    // Asegurar que el resultado coincide con los datos esperados
    $this->assertEquals($mockedData, $result);
}
```



testVerify	Verifica que el método verify devuelve los datos correctos de una reserva cuando hay superposición de fechas.
<p>Codigo:</p> <pre>public function testVerify() { // Preparar los datos esperados \$mockedData = ['id' => 1, 'fecha_prestamo' => '2022-05-10', 'fecha_devolucion' => '2022-05-15']; // Suponiendo un resultado de reserva superpuesta // Configurar el comportamiento esperado del mock de Query para el método select \$this->mockQuery->shouldReceive('select') ->once() ->andReturn(\$mockedData); // Parámetros para verificar la reserva \$desde = '2022-05-12'; \$hasta = '2022-05-14'; \$id_veh = 1; // Instanciar el modelo y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->verify(\$desde, \$hasta, \$id_veh); // Asegurar que el resultado coincide con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	
testActualizarVehiculo	Verifica que el método actualizarVehiculo actualiza correctamente el estado de un vehículo y devuelve "ok".
<p>Codigo:</p> <pre>public function testActualizarVehiculo() { // Preparar los datos para la actualización del vehículo \$estado = 0; // Estado a actualizar \$vehiculoId = 1; // ID del vehículo // Configurar el comportamiento esperado del mock de Query para el método save \$this->mockQuery->shouldReceive('save') ->once() ->andReturn(1); // Suponiendo que se actualiza correctamente</pre>	



<pre>// Instanciar el modelo y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->actualizarVehiculo(\$estado, \$vehiculoId); // Asegurar que el resultado coincide con 'ok' \$this->assertEquals('ok', \$result); }</pre>	
testActualizarVehiculoError	Verifica que el método actualizarVehiculo devuelve "error" cuando la actualización del estado de un vehículo falla.
<p>Codigo:</p> <pre>public function testActualizarVehiculoError() { // Datos simulados para la actualización del vehículo \$estado = 1; \$id = 1; // Configurar el mock para el método save en la actualización de vehículo \$this->mockQuery->shouldReceive('save') ->once() ->andReturn(0); // Simular que hubo un error en la actualización // Instanciar AlquilerModel y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->actualizarVehiculo(\$estado, \$id); // Verificar que se retorna el resultado esperado en caso de error en la actualización ('error') \$this->assertEquals('error', \$result); }</pre>	
testProcesarEntrega	Verifica que el método procesarEntrega actualiza correctamente el estado de una entrega de alquiler y devuelve "ok".
<p>Codigo:</p> <pre>public function testProcesarEntrega() { // Preparar los datos para procesar la entrega \$estado = 1; // Estado a actualizar \$alquilerId = 1; // ID del alquiler // Configurar el comportamiento esperado del mock de Query para el método save</pre>	



<pre>\$this->mockQuery->shouldReceive('save') ->once() ->andReturn(1); // Suponiendo que se actualiza correctamente // Instanciar el modelo y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->procesarEntrega(\$estado, \$alquilerId); // Asegurar que el resultado coincide con 'ok' \$this->assertEquals('ok', \$result); }</pre>	
testProcesarEntregaError	Verifica que el método procesarEntrega devuelve "error" cuando la actualización del estado de una entrega de alquiler falla.
Codigo: <pre>public function testProcesarEntregaError() { // Datos simulados para procesar la entrega \$estado = 2; // Nuevo estado \$id = 1; // ID del registro de alquiler a procesar // Configurar el mock para el método save en la actualización del estado de alquiler \$this->mockQuery->shouldReceive('save') ->once() ->andReturn(0); // Simular que hubo un error en la actualización // Instanciar AlquilerModel y llamar al método que queremos probar \$alquilerModel = new AlquilerModel(\$this->mockQuery); \$result = \$alquilerModel->procesarEntrega(\$estado, \$id); // Verificar que se retorna el resultado esperado en caso de error en la actualización ('error') \$this->assertEquals('error', \$result); }</pre>	
testVerPrestamo	Verifica que el método verPrestamo devuelve los datos correctos de un prestamo cuando se llama al método select con el ID del préstamo.
Codigo: <pre>public function testVerPrestamo() { // Preparar los datos esperados</pre>	



```
$mockedData = [  
    'id' => 1,  
    'fecha_prestamo' => '2022-05-10',  
    'fecha_devolucion' => '2022-05-15',  
    'dni' => '12345678',  
    'nombre' => 'Cliente A',  
    'telefono' => '123456789',  
    'direccion' => 'Calle A',  
    'placa' => 'ABC123',  
    'modelo' => 'Modelo X',  
    'documento' => 'Documento A',  
    'tipo' => 'Tipo A',  
];
```

- Test de CarsModel.php

Titulo del Test	Descripción
testGetVehiculos	Este test verifica que el método getVehiculos del modelo CarsModel devuelve los vehículos activos correctamente, simulando el comportamiento de la consulta SQL esperada.

Codigo:

```
public function testGetVehiculos()  
{  
    // Preparar los datos esperados  
    $mockedData = [  
        ['id' => 1, 'placa' => 'ABC123', 'id_tipo' => 1, 'id_marca' => 1, 'estado' =>  
1],  
        ['id' => 2, 'placa' => 'XYZ789', 'id_tipo' => 2, 'id_marca' => 2, 'estado' =>  
1],  
    ];  
  
    // Estado para filtrar vehículos activos (estado = 1)  
    $estado = 1;  
  
    // Configurar el comportamiento esperado del mock de Query para el método selectAll  
    $this->mockQuery->shouldReceive('selectAll')  
        ->once()  
        ->with("SELECT v.*, m.marca, t.tipo FROM vehiculos v INNER JOIN marcas m ON  
v.id_marca = m.id INNER JOIN tipos t ON v.id_tipo = t.id WHERE v.estado != $estado")  
        ->andReturn($mockedData);  
  
    // Instanciar el modelo y llamar al método que queremos probar
```



<pre>\$carsModel = new CarsModel(\$this->mockQuery); \$result = \$carsModel->getVehiculos(\$estado); // Asegurar que el resultado coincide con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	
testGetVehiculo	Este test comprueba que el método getVehiculo del modelo CarsModel devuelve la información del vehículo específico solicitado, simulando el comportamiento de la consulta SQL esperada.
<p>Codigo:</p> <pre>public function testGetVehiculo() { // Preparar los datos esperados \$mockedData = ['id' => 1, 'placa' => 'ABC123', 'id_tipo' => 1, 'id_marca' => 1, 'estado' => 1]; // Configurar el comportamiento esperado del mock de Query para el método select \$this->mockQuery->shouldReceive('select') ->once() ->andReturn(\$mockedData); // ID del vehículo a consultar \$vehiculoId = 1; // Instanciar el modelo y llamar al método que queremos probar \$carsModel = new CarsModel(\$this->mockQuery); \$result = \$carsModel->getVehiculo(\$vehiculoId); // Asegurar que el resultado coincide con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	

- Test de ClientesModel.php

Titulo del Test	Descripción
testGetClientes	Verifica que el método getClientes del modelo ClientesModel devuelva correctamente los clientes activos.
<p>Codigo:</p> <pre>public function testGetClientes() { </pre>	



<pre>\$mockedData = [['id' => 1, 'nombre' => 'Cliente 1', 'direccion' => 'Dirección 1'], ['id' => 2, 'nombre' => 'Cliente 2', 'direccion' => 'Dirección 2'],]; // Configurar el mock para selectAll en getClientes \$this->mockQuery->shouldReceive('selectAll') ->once() ->with("SELECT * FROM clientes WHERE estado = 1") ->andReturn(\$mockedData); \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->getClientes(1); \$this->assertEquals(\$mockedData, \$result); }</pre>	
testBuscarCliente	Comprueba que el método buscarCliente del modelo ClientesModel retorne los clientes que coincidan con el nombre buscado.
<p><u>Codigo:</u></p> <pre>public function testBuscarCliente() { \$mockedData = [['id' => 1, 'nombre' => 'Cliente 1', 'direccion' => 'Dirección 1'], ['id' => 2, 'nombre' => 'Cliente 2', 'direccion' => 'Dirección 2'],]; // Configurar el mock para selectAll en buscarCliente \$this->mockQuery->shouldReceive('selectAll') ->once() ->with("SELECT id, nombre, direccion FROM clientes WHERE nombre LIKE '%valor_buscado%' AND estado = 1") ->andReturn(\$mockedData); \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->buscarCliente('valor_buscado'); \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRegistrarCliente_Exit	Confirma que el método registrarCliente del modelo ClientesModel registre un nuevo cliente correctamente cuando no existe previamente.



Codigo:

```
public function testRegistrarCliente_Exito()
{
    // Configurar el mock para el método select en la verificación de existencia de
    cliente
    $this->mockQuery->shouldReceive('select')
        ->andReturn([]); // Cliente no existe

    // Configurar el mock para el método save en el registro de cliente
    $this->mockQuery->shouldReceive('save')
        ->once()
        ->andReturn(1); // Éxito en la inserción

    $nombre = 'Nuevo Cliente';
    $dni = '12345678A';
    $telefono = '987654321';
    $direccion = 'Calle Principal';

    $clientesModel = new ClientesModel($this->mockQuery);
    $result = $clientesModel->registrarCliente($dni, $nombre, $telefono, $direccion);

    $this->assertEquals('ok', $result); // Verificar que se retorna 'ok' en caso de éxito
}
```

testRegistrarCliente_Error

Verifica que el método registrarCliente del modelo ClientesModel maneje adecuadamente el caso de error al intentar registrar un cliente nuevo.

Codigo:

```
public function testRegistrarCliente_Error()
{
    // Configurar el mock para el método select en la verificación de existencia de
    cliente
    $this->mockQuery->shouldReceive('select')
        ->andReturn([]); // Simulamos que el cliente no existe

    // Configurar el mock para el método save en el registro de cliente
    $this->mockQuery->shouldReceive('save')
        ->once()
        ->andReturn(0); // Simulamos que hubo un error en la inserción

    $nombre = 'Nuevo Cliente';
    $dni = '12345678A';
    $telefono = '987654321';
```



<pre>\$direccion = 'Calle Principal'; \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->registrarCliente(\$dni, \$nombre, \$telefono, \$direccion); \$this->assertEquals('error', \$result); // Verificar que se retorna 'error' cuando falla la inserción }</pre>	
testRegistrarClienteExistente	Comprueba que el método registrarCliente del modelo ClientesModel detecte correctamente cuando se intenta registrar un cliente que ya existe en la base de datos.
<p>Codigo:</p> <pre>public function testRegistrarClienteExistente() { \$nombre = 'Cliente Existente'; \$dni = '12345678A'; \$telefono = '987654321'; \$direccion = 'Calle Principal'; // Configurar el mock para el método select en la verificación de existencia de cliente // Simulamos que el cliente ya existe en la base de datos \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM clientes WHERE nombre = '\$nombre'") ->andReturn(['id' => 1, 'nombre' => \$nombre]); \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->registrarCliente(\$dni, \$nombre, \$telefono, \$direccion); // Verificar que se retorna 'existe' si el cliente ya existe en la base de datos \$this->assertEquals('existe', \$result); }</pre>	
testModificarCliente_success	Asegura que el método modificarCliente del modelo ClientesModel actualice la información de un cliente existente correctamente cuando la operación tiene éxito.
<p>Codigo:</p> <pre>public function testModificarCliente_success() { \$mockedDni = '12345678A';</pre>	



<pre>\$mockedNombre = 'Nuevo Nombre'; \$mockedTelefono = '987654321'; \$mockedDireccion = 'Calle Principal'; \$mockedId = 1; // Configuración del mock para caso de éxito \$this->mockQuery->shouldReceive('save') ->once() ->with("UPDATE clientes SET dni = ?, nombre = ?, telefono = ?, direccion = ? WHERE id = ?", [\$mockedDni, \$mockedNombre, \$mockedTelefono, \$mockedDireccion, \$mockedId]) ->andReturn(1); \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->modificarCliente(\$mockedDni, \$mockedNombre, \$mockedTelefono, \$mockedDireccion, \$mockedId); // Verificar que el método devuelve 'modificado' en caso de éxito \$this->assertEquals('modificado', \$result); }</pre>	
testModificarCliente_failure	Verifica que el método modificarCliente del modelo ClientesModel maneje adecuadamente el caso de fallo al intentar modificar la información de un cliente existente.
<p><u>Codigo:</u></p> <pre>public function testModificarCliente_failure() { \$mockedDni = '12345678A'; \$mockedNombre = 'Nuevo Nombre'; \$mockedTelefono = '987654321'; \$mockedDireccion = 'Calle Principal'; \$mockedId = 1; // Configuración del mock para caso de error \$this->mockQuery->shouldReceive('save') ->once() ->with("UPDATE clientes SET dni = ?, nombre = ?, telefono = ?, direccion = ? WHERE id = ?", [\$mockedDni, \$mockedNombre, \$mockedTelefono, \$mockedDireccion, \$mockedId]) ->andReturn(0); \$clientesModel = new ClientesModel(\$this->mockQuery);</pre>	



<pre> \$result = \$clientesModel->modificarCliente(\$mockedDni, \$mockedNombre, \$mockedTelefono, \$mockedDireccion, \$mockedId); // Verificar que el método devuelve 'error' en caso de error \$this->assertEquals('error', \$result); } }</pre>	
testEditarCli	Confirma que el método editarCli del modelo ClientesModel obtenga la información de un cliente específico correctamente.
<p>Codigo:</p> <pre>public function testEditarCli() { \$mockedId = 1; \$mockedData = ['id' => 1, 'nombre' => 'Cliente Ejemplo', 'direccion' => 'Calle Principal']; // Configurar el mock para select en editarCli \$this->mockQuery->shouldReceive('select') ->once() ->with("SELECT * FROM clientes WHERE id = \$mockedId") ->andReturn(\$mockedData); \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->editarCli(\$mockedId); \$this->assertEquals(\$mockedData, \$result); }</pre>	
testAccionCli	Asegura que el método accionCli del modelo ClientesModel maneje correctamente la acción de cambiar el estado (activar/desactivar) de un cliente.
<p>Codigo:</p> <pre>public function testAccionCli() { \$mockedEstado = 0; \$mockedId = 1; // Configurar el mock para save en accionCli \$this->mockQuery->shouldReceive('save') ->once() ->with("UPDATE clientes SET estado = ? WHERE id = ?", [\$mockedEstado, \$mockedId]) ->andReturn(1); // Suponiendo que la actualización tiene éxito }</pre>	



<pre>\$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->accionCli(\$mockedEstado, \$mockedId); \$this->assertEquals(1, \$result); // Verificar que el resultado sea 1 (éxito) }</pre>	
testModificarPass	Verifica que el método modificarPass del modelo ClientesModel actualice correctamente la contraseña de un cliente.
Codigo: <pre>public function testModificarPass() { \$mockedClave = 'nuevaclave123'; \$mockedId = 1; // Configurar el mock para save en modificarPass \$this->mockQuery->shouldReceive('save') ->once() ->with("UPDATE clientes SET clave = ? WHERE id = ?", [\$mockedClave, \$mockedId]) ->andReturn(1); // Suponiendo que la actualización tiene éxito \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->modificarPass(\$mockedClave, \$mockedId); \$this->assertEquals(1, \$result); // Verificar que el resultado sea 1 (éxito) }</pre>	
testModificarDato_ActualizacionExitosa	Comprueba que el método modificarDato del modelo ClientesModel actualice los datos de un cliente correctamente cuando la operación tiene éxito.
Codigo: <pre>public function testModificarDato_ActualizacionExitosa() { \$nombre = 'Nuevo Nombre'; \$dni = '12345678A'; \$correo = 'nuevo@correo.com'; \$tel = '987654321'; \$dir = 'Calle Principal'; \$id = 1; // Configurar el mock para el método save en la modificación de datos del cliente \$this->mockQuery->shouldReceive('save') ->with("UPDATE clientes SET nombre=?, dni=?, correo=?, telefono=?, direccion=? WHERE id=?",</pre>	



<pre>[\$nombre, \$dni, \$correo, \$tel, \$dir, \$id]) ->andReturn(1); // Simulamos que la actualización tiene éxito \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->modificarDato(\$nombre, \$dni, \$correo, \$tel, \$dir, \$id); // Verificar que se retorna 1 cuando la actualización tiene éxito \$this->assertEquals(1, \$result); }</pre>	
testModificarDato_ActualizacionNoExitosa	Verifica que el método modificarDato del modelo ClientesModel maneje adecuadamente el caso de fallo al intentar actualizar los datos de un cliente.
<p>Codigo:</p> <pre>public function testModificarDato_ActualizacionNoExitosa() { \$nombre = 'Nuevo Nombre'; \$dni = '12345678A'; \$correo = 'nuevo@correo.com'; \$tel = '987654321'; \$dir = 'Calle Principal'; \$id = 1; // Configurar el mock para el método save en la modificación de datos del cliente \$this->mockQuery->shouldReceive('save') ->with("UPDATE clientes SET nombre=?, dni=?, correo=?, telefono=?, direccion=? WHERE id=?", [\$nombre, \$dni, \$correo, \$tel, \$dir, \$id]) ->andReturn(0); // Simulamos que la actualización no tiene éxito \$clientesModel = new ClientesModel(\$this->mockQuery); \$result = \$clientesModel->modificarDato(\$nombre, \$dni, \$correo, \$tel, \$dir, \$id); // Verificar que se retorna 0 cuando la actualización no tiene éxito \$this->assertEquals(0, \$result); }</pre>	

- Test de ContactModel.php

Titulo del Test	Descripción
testGetEmpresa	Verifica que el método getEmpresa del modelo ContactModel obtenga correctamente los datos de configuración de la empresa mediante una consulta simulada.



Codigo:

```
public function testGetEmpresa()
{
    // Datos simulados que se esperan retornar del método select
    $mockedData = [
        ['id' => 1, 'nombre' => 'Empresa A', 'direccion' => 'Calle Principal', 'telefono'
=> '123456789'],
    ];

    // Configurar el mock para el método select en la consulta de empresa
    $this->mockQuery->shouldReceive('select')
        ->with("SELECT * FROM configuracion")
        ->andReturn($mockedData);

    // Instanciar ContactModel con el mock de Query
    $contactModel = new ContactModel($this->mockQuery);

    // Llamar al método getEmpresa y almacenar el resultado
    $result = $contactModel->getEmpresa();

    // Verificar que el resultado retornado sea el mismo que $mockedData
    $this->assertEquals($mockedData, $result);
}
```

- Test de DashboardModel.php

Titulo del Test	Descripción
testGetDatos	Verifica que el método getDatos del modelo DashboardModel obtenga correctamente el número total de usuarios activos mediante una consulta simulada.

Codigo:

```
public function testGetDatos()
{
    // Datos simulados que se esperan retornar del método select
    $mockedData = [['total' => 10]];

    // Configurar el mock para el método select en la consulta de getDatos
    $this->mockQuery->shouldReceive('select')
        ->with("SELECT COUNT(*) AS total FROM usuarios WHERE estado = 1")
        ->andReturn($mockedData);

    // Instanciar DashboardModel con el mock de Query
    $dashboardModel = new DashboardModel($this->mockQuery);
}
```



<pre>// Llamar al método getDatos y almacenar el resultado \$result = \$dashboardModel->getDatos('usuarios'); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRentas	Verifica que el método rentas del modelo DashboardModel obtenga correctamente las rentas acumuladas durante un período específico mediante una consulta simulada.
<p><u>Codigo:</u></p> <pre>public function testRentas() { // Datos simulados que se esperan retornar del método select \$mockedData = [['ene' => 100, 'feb' => 200, 'mar' => 150]]; // Configurar el mock para el método select en la consulta de rentas \$this->mockQuery->shouldReceive('select') ->with(M::on(function (\$sql) { // Verificar que el SQL contenga las fechas esperadas return strpos(\$sql, "BETWEEN '2024-01-01' AND '2024-12-31'") !== false; }))) ->andReturn(\$mockedData); // Instanciar DashboardModel con el mock de Query \$dashboardModel = new DashboardModel(\$this->mockQuery); // Llamar al método rentas y almacenar el resultado \$result = \$dashboardModel->rentas('2024-01-01', '2024-12-31'); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRentasSemana	Verifica que el método rentasSemana del modelo DashboardModel obtenga correctamente las rentas de la semana actual mediante una consulta simulada.
<p><u>Codigo:</u></p> <pre>public function testRentasSemana() { // Datos simulados que se esperan retornar del método selectAll \$mockedData = [['fecha' => '2024-06-24', 'total' => 50],]</pre>	



```
        ['fecha' => '2024-06-25', 'total' => 100],
        ['fecha' => '2024-06-26', 'total' => 80],
    ];

    // Configurar el mock para el método selectAll en la consulta de rentasSemana
    $this->mockQuery->shouldReceive('selectAll')
        ->with(M::on(function ($sql) {
            // Verificar que el SQL incluya la semana actual
            return strpos($sql, "WEEK(fecha_prestamo) = WEEK(CURDATE())") !== false;
        })))
        ->andReturn($mockedData);

    // Instanciar DashboardModel con el mock de Query
    $dashboardModel = new DashboardModel($this->mockQuery);

    // Llamar al método rentasSemana y almacenar el resultado
    $result = $dashboardModel->rentasSemana();

    // Verificar que el resultado retornado sea el mismo que $mockedData
    $this->assertEquals($mockedData, $result);
}
```

- Test de DocumentosModel.php

Titulo del Test	Descripción
testGetDocumentos	Verifica que el método getDocumentos del modelo DocumentosModel obtenga correctamente todos los documentos activos mediante una consulta simulada.

Codigo:

```
public function testGetDocumentos()
{
    // Datos simulados que se esperan retornar del método selectAll
    $mockedData = [
        ['id' => 1, 'documento' => 'Documento A', 'estado' => 1],
        ['id' => 2, 'documento' => 'Documento B', 'estado' => 1],
    ];

    // Configurar el mock para el método selectAll en la consulta de getDocumentos
    $this->mockQuery->shouldReceive('selectAll')
        ->with("SELECT * FROM documentos WHERE estado = 1")
        ->andReturn($mockedData);

    // Instanciar DocumentosModel con el mock de Query
    $documentosModel = new DocumentosModel($this->mockQuery);
}
```



<pre>// Llamar al método getDocumentos y almacenar el resultado \$result = \$documentosModel->getDocumentos(1); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRegistrarDoc	Verifica que el método registrarDoc del modelo DocumentosModel registre un nuevo documento correctamente mediante consultas simuladas de verificación e inserción.
<p>Codigo:</p> <pre>public function testRegistrarDoc() { // Documento a registrar \$documento = 'Nuevo Documento'; // Configurar el mock para el método select en la consulta de verificar existencia \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM documentos WHERE documento = '\$documento'") ->andReturn([]); // Simular que el documento no existe // Configurar el mock para el método save en la consulta de inserción \$this->mockQuery->shouldReceive('save') ->once() ->andReturn(1); // Éxito en la inserción // Instanciar DocumentosModel con el mock de Query \$documentosModel = new DocumentosModel(\$this->mockQuery); // Llamar al método registrarDoc y almacenar el resultado \$result = \$documentosModel->registrarDoc(\$documento); // Verificar que el resultado sea "ok" (éxito en la inserción) \$this->assertEquals('ok', \$result); }</pre>	
testRegistrarDoc_Error	Verifica que el método registrarDoc del modelo DocumentosModel maneje correctamente el escenario de error durante la inserción de un nuevo documento.
<p>Codigo:</p> <pre>public function testRegistrarDoc_Error() { </pre>	



<pre>// Configurar el mock para el método select en la verificación de existencia de documento \$this->mockQuery->shouldReceive('select') ->andReturn([]); // Documento no existe // Configurar el mock para el método save en el registro de documento \$this->mockQuery->shouldReceive('save') ->once() ->andReturn(0); // Error en la inserción \$documento = '12345678A'; \$documentosModel = new DocumentosModel(\$this->mockQuery); \$result = \$documentosModel->registrarDoc(\$documento); \$this->assertEquals('error', \$result); // Verificar que se retorna 'error' cuando falla la inserción }</pre>	
testRegistrarDocDocumentoExistente	Verifica que el método registrarDoc del modelo DocumentosModel maneje correctamente el escenario donde se intenta registrar un documento que ya existe.
<p>Codigo:</p> <pre>public function testRegistrarDocDocumentoExistente() { // Documento que ya existe en la base de datos \$documentoExistente = 'Documento A'; // Configurar el mock para el método select en la consulta de verificar existencia \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM documentos WHERE documento = '\$documentoExistente'") ->andReturn(['id' => 1, 'documento' => \$documentoExistente, 'estado' => 1]); // Simular que el documento ya existe // Instanciar DocumentosModel con el mock de Query \$documentosModel = new DocumentosModel(\$this->mockQuery); // Llamar al método registrarDoc con el documento existente y almacenar el resultado \$result = \$documentosModel->registrarDoc(\$documentoExistente); // Verificar que el resultado sea "existe" \$this->assertEquals('existe', \$result); }</pre>	



<pre>}</pre>	
testModificarDoc	Verifica que el método modificarDoc del modelo DocumentosModel modifique un documento existente correctamente mediante una consulta simulada.
<p>Codigo:</p> <pre>public function testModificarDoc() { // Documento modificado y su ID \$documento = 'Documento Modificado'; \$id = 1; // Configurar el mock para el método save en la consulta de actualización \$this->mockQuery->shouldReceive('save') ->with("UPDATE documentos SET documento = ? WHERE id = ?", [\$documento, \$id]) ->once() ->andReturn(1); // Éxito en la modificación // Instanciar DocumentosModel con el mock de Query \$documentosModel = new DocumentosModel(\$this->mockQuery); // Llamar al método modificarDoc y almacenar el resultado \$result = \$documentosModel->modificarDoc(\$documento, \$id); // Verificar que el resultado sea "modificado" (éxito en la modificación) \$this->assertEquals('modificado', \$result); }</pre>	
testModificarDocError	Verifica que el método modificarDoc del modelo DocumentosModel maneje correctamente el escenario de error durante la modificación de un documento.
<p>Codigo:</p> <pre>public function testModificarDocError() { // Documento y ID para la modificación \$documento = 'Documento Modificado'; \$id = 1; // ID existente en la base de datos // Configurar el mock para el método save en la actualización del documento \$this->mockQuery->shouldReceive('save') ->once() // Esperamos una sola llamada al método save ->with("UPDATE documentos SET documento = ? WHERE id = ?", [\$documento, \$id]) ->andReturn(0); // Simular que la actualización del documento falló</pre>	



<pre>// Instanciar DocumentosModel con el mock de Query \$documentosModel = new DocumentosModel(\$this->mockQuery); // Llamar al método modificarDoc y almacenar el resultado \$result = \$documentosModel->modificarDoc(\$documento, \$id); // Verificar que el resultado sea "error" \$this->assertEquals('error', \$result); }</pre>	
testEditarDoc	Verifica que el método editarDoc del modelo DocumentosModel obtenga correctamente los datos de un documento específico mediante una consulta simulada.
<p>Codigo:</p> <pre>public function testEditarDoc() { // ID del documento a editar \$id = 1; // Datos simulados que se esperan retornar del método select \$mockedData = ['id' => 1, 'documento' => 'Documento A', 'estado' => 1]; // Configurar el mock para el método select en la consulta de editarDoc \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM documentos WHERE id = \$id") ->andReturn(\$mockedData); // Instanciar DocumentosModel con el mock de Query \$documentosModel = new DocumentosModel(\$this->mockQuery); // Llamar al método editarDoc y almacenar el resultado \$result = \$documentosModel->editarDoc(\$id); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testAccionDoc	Verifica que el método accionDoc del modelo DocumentosModel modifique el estado de un documento correctamente mediante una consulta simulada.
<p>Codigo:</p> <pre>public function testAccionDoc() { // Estado y ID del documento a modificar \$estado = 0;</pre>	



```
$id = 1;

// Configurar el mock para el método save en la consulta de accionDoc
$this->mockQuery->shouldReceive('save')
    ->with("UPDATE documentos SET estado = ? WHERE id = ?", [$estado, $id])
    ->once()
    ->andReturn(1); // Éxito en la actualización del estado
// Instanciar DocumentosModel con el mock de Query
$documentosModel = new DocumentosModel($this->mockQuery);
// Llamar al método accionDoc y almacenar el resultado
$result = $documentosModel->accionDoc($estado, $id);

// Verificar que el resultado retornado sea 1 (éxito en la actualización del estado)
$this->assertEquals(1, $result);
}
```

- Test de EmpresaModel.php

Titulo del Test	Descripción
testGetEmpresa	Verifica que el método getEmpresa del modelo EmpresaModel obtenga correctamente los datos de la empresa mediante una consulta simulada.

Codigo:

```
public function testGetEmpresa()
{
    // Datos simulados que se esperan retornar del método select
    $mockedData = [
        'id' => 1,
        'ruc' => '123456789',
        'nombre' => 'Empresa A',
        'telefono' => '987654321',
        'correo' => 'empresa@example.com',
        'direccion' => 'Calle Principal',
        'mensaje' => 'Bienvenido a nuestra empresa',
        'logo' => 'logo.jpg',
    ];

    // Configurar el mock para el método select en la consulta de empresa
    $this->mockQuery->shouldReceive('select')
        ->once()
        ->with("SELECT * FROM configuracion")
        ->andReturn($mockedData);

    // Instanciar el modelo y llamar al método que queremos probar
}
```



<pre>\$empresaModel = new EmpresaModel(\$this->mockQuery); \$result = \$empresaModel->getEmpresa(); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testModificar	Verifica que el método modificar del modelo EmpresaModel actualice correctamente los datos de la empresa mediante una consulta simulada.
<p>Codigo:</p> <pre>public function testModificar() { // Datos simulados para la modificación \$ruc = '123456789'; \$nombre = 'Empresa B'; \$tel = '987654321'; \$correo = 'empresa@example.com'; \$dir = 'Calle Secundaria'; \$mensaje = 'Bienvenido a nuestra nueva empresa'; \$img = 'logo.jpg'; \$id = 1; // Configurar el mock para el método save en la actualización de configuracion \$this->mockQuery->shouldReceive('save') ->once() ->with("UPDATE configuracion SET ruc=?, nombre = ?, telefono =?, correo=?, direccion=?, mensaje=?, logo = ? WHERE id=?", [\$ruc, \$nombre, \$tel, \$correo, \$dir, \$mensaje, \$img, \$id]) ->andReturn(1); // Simular que la actualización fue exitosa // Instanciar el modelo y llamar al método que queremos probar \$empresaModel = new EmpresaModel(\$this->mockQuery); \$result = \$empresaModel->modificar(\$ruc, \$nombre, \$tel, \$correo, \$dir, \$mensaje, \$img, \$id); // Verificar que el resultado retornado sea "ok" \$this->assertEquals('ok', \$result); }</pre>	



testModificarError	Verifica que el método modificar del modelo EmpresaModel maneje correctamente el escenario de error durante la actualización de datos.
<p>Codigo:</p> <pre>public function testModificarError() { // Datos simulados para la modificación \$ruc = '123456789'; \$nombre = 'Empresa B'; \$tel = '987654321'; \$correo = 'empresa@example.com'; \$dir = 'Calle Secundaria'; \$mensaje = 'Bienvenido a nuestra nueva empresa'; \$img = 'logo.jpg'; \$id = 1; // Configurar el mock para el método save en la actualización de configuracion \$this->mockQuery->shouldReceive('save') ->once() ->with("UPDATE configuracion SET ruc=?, nombre = ?, telefono =?, correo=?, direccion=?, mensaje=?, logo = ? WHERE id=?", [\$ruc, \$nombre, \$tel, \$correo, \$dir, \$mensaje, \$img, \$id]) ->andReturn(0); // Simular que la actualización falló // Instanciar el modelo y llamar al método que queremos probar \$empresaModel = new EmpresaModel(\$this->mockQuery); \$result = \$empresaModel->modificar(\$ruc, \$nombre, \$tel, \$correo, \$dir, \$mensaje, \$img, \$id); // Verificar que el resultado retornado sea "error" \$this->assertEquals('error', \$result); }</pre>	

- Test de LoginModel.php

Título del Test	Descripción
testVerifyWithOtherTable	Verifica que el método verify del modelo LoginModel valide correctamente las credenciales del usuario contra otra tabla mediante una consulta simulada.
Codigo:	



```
public function testVerifyWithOtherTable()
{
    // Datos simulados que se esperan retornar del método select
    $mockedData = [['id' => 1, 'correo' => 'admin@example.com', 'clave' =>
'hashed_password', 'estado' => 1]];

    // Configurar el mock para el método select en la consulta de verify
    $this->mockQuery->shouldReceive('select')
->with("SELECT * FROM otra_tabla WHERE correo = 'admin@example.com' AND clave =
'hashed_password' AND estado = 1")
->andReturn($mockedData);

    // Instanciar LoginModel con el mock de Query
    $loginModel = new LoginModel($this->mockQuery);

    // Llamar al método verify y almacenar el resultado
    $result = $loginModel->verify('otra_tabla', 'admin@example.com', 'hashed_password');

    // Verificar que el resultado retornado sea el mismo que $mockedData
    $this->assertEquals($mockedData, $result);
}
```

- Test de MajorModel.php

Titulo del Test	Descripción
testVerify	Verifica que el método verify del modelo MajorModel obtenga correctamente los datos de un cliente específico mediante una consulta simulada.
Codigo: <pre>public function testVerify() { // Datos simulados que se esperan retornar del método select \$item = 'correo'; \$valor = 'test@example.com'; \$mockedData = [['id' => 1, 'nombre' => 'Test Cliente', 'correo' => 'test@example.com', 'codphone' => '123', 'telefono' => '456', 'direccion' => 'Calle Principal', 'clave' => 'hashed_password']]; // Configurar el mock para el método select en la consulta de verify \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM clientes WHERE correo = 'test@example.com'") ->andReturn(\$mockedData); // Instanciar MajorModel con el mock de Query</pre>	



```
$majorModel = new MajorModel($this->mockQuery);

// Llamar al método verify y almacenar el resultado
$result = $majorModel->verify($item, $valor);

// Verificar que el resultado retornado sea el mismo que $mockedData
$this->assertEquals($mockedData, $result);
}
```

testRegister

Verifica que el método register del modelo MajorModel registre un nuevo cliente correctamente mediante una consulta de inserción simulada.

Codigo:

```
public function testRegister()
{
    // Datos simulados para la inserción
    $nombre = 'Nuevo Cliente';
    $correo = 'nuevo@example.com';
    $codphone = '234';
    $telefono = '789';
    $direccion = 'Avenida Principal';
    $clave = 'hashed_password';

    // Configurar el mock para el método insertar en la consulta de register
    $this->mockQuery->shouldReceive('insertar')
        ->with(
            "INSERT INTO clientes (nombre, correo, codphone, telefono, direccion, clave)
VALUES (?, ?, ?, ?, ?, ?)",
            [$nombre, $correo, $codphone, $telefono, $direccion, $clave]
        )
        ->andReturn(true); // Supongamos que devuelve verdadero (éxito)

    // Instanciar MajorModel con el mock de Query
    $majorModel = new MajorModel($this->mockQuery);

    // Llamar al método register y almacenar el resultado
    $result = $majorModel->register($nombre, $correo, $codphone, $telefono, $direccion,
$clave);

    // Verificar que el resultado retornado sea verdadero (éxito)
    $this->assertTrue($result);
}
```



- Test de MarcasModel.php

Titulo del Test	Descripción
testGetMarcas	Verifica que el método getMarcas retorna las marcas activas de la base de datos según el estado proporcionado.
Codigo: <pre>public function testGetMarcas() { // Datos simulados que se esperan retornar del método selectAll \$estado = 1; \$mockedData = [['id' => 1, 'marca' => 'Marca A', 'estado' => 1], ['id' => 2, 'marca' => 'Marca B', 'estado' => 1],]; // Configurar el mock para el método selectAll en la consulta de getMarcas \$this->mockQuery->shouldReceive('selectAll') ->with("SELECT * FROM marcas WHERE estado = \$estado") ->andReturn(\$mockedData); // Instanciar MarcasModel con el mock de Query \$marcasModel = new MarcasModel(); // Llamar al método getMarcas y almacenar el resultado \$result = \$marcasModel->getMarcas(\$estado); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRegistrarMarca	Verifica que el método registrarMarca inserta una nueva marca en la base de datos cuando no existe una marca con el mismo nombre.
Codigo: <pre>public function testRegistrarMarca() { // Datos simulados para la inserción \$marcaNueva = 'Nueva Marca'; // Configurar el mock para el método select y save en la consulta de registrarMarca \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM marcas WHERE marca = '\$marcaNueva'") ->andReturn([]); // Simular que la marca no existe</pre>	



```
$this->mockQuery->shouldReceive('save')
    ->with("INSERT INTO marcas(marca) VALUES (?)", [$marcaNueva])
    ->andReturn(1); // Supongamos que devuelve 1 (éxito)

// Instanciar MarcasModel con el mock de Query
$marcasModel = new MarcasModel();

// Llamar al método registrarMarca y almacenar el resultado
$result = $marcasModel->registrarMarca($marcaNueva);

// Verificar que el resultado retornado sea "ok"
$this->assertEquals('ok', $result);
}
```

testRegistrarMarcaExistente

Verifica que el método registrarMarca retorna "existe" cuando se intenta registrar una marca que ya existe en la base de datos.

Codigo:

```
public function testRegistrarMarcaExistente()
{
    // Marca que ya existe en la base de datos
    $marcaExistente = 'MarcaExistente';

    // Configurar el mock para el método select en la consulta de registrarMarca
    $this->mockQuery->shouldReceive('select')
        ->with("SELECT * FROM marcas WHERE marca = '$marcaExistente'")
        ->andReturn([[ 'id' => 1, 'marca' => $marcaExistente ]]); // Simular que la marca
    existe

    // No se debe llamar a save, así que no configuramos expectativa para save

    // Instanciar MarcasModel con el mock de Query
    $marcasModel = new MarcasModel();

    // Llamar al método registrarMarca con la marca existente y almacenar el resultado
    $result = $marcasModel->registrarMarca($marcaExistente);

    // Verificar que el resultado retornado sea "existe"
    $this->assertEquals('existe', $result);
}
```



testRegistrarMarcaError	Verifica que el método registrarMarca retorna "error" cuando ocurre un error al intentar insertar una nueva marca en la base de datos.
<p>Codigo:</p> <pre>public function testRegistrarMarcaError() { // Marca a registrar \$marcaNueva = 'Nueva Marca'; // Configurar el mock para el método select y save en la consulta de registrarMarca \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM marcas WHERE marca = '\$marcaNueva'") ->andReturn([]); // Simular que la marca no existe // Configurar el mock para simular un error en la inserción \$this->mockQuery->shouldReceive('save') ->with("INSERT INTO marcas(marca) VALUES (?)", [\$marcaNueva]) ->andReturn(0); // Simular que devuelve 0 (error) // Instanciar MarcasModel con el mock de Query \$marcasModel = new MarcasModel(); // Llamar al método registrarMarca y almacenar el resultado \$result = \$marcasModel->registrarMarca(\$marcaNueva); // Verificar que el resultado retornado sea "error" \$this->assertEquals('error', \$result); }</pre>	
testModificarMarca	Verifica que el método modificarMarca actualiza correctamente una marca existente en la base de datos.
<p>Codigo:</p> <pre>public function testModificarMarca() { // Datos simulados para la modificación \$marcaModificada = 'Marca Modificada'; \$idMarca = 1; // Configurar el mock para el método save en la consulta de modificarMarca \$this->mockQuery->shouldReceive('save') ->with("UPDATE marcas SET marca = ? WHERE id = ?", [\$marcaModificada, \$idMarca]) ->andReturn(1); // Supongamos que devuelve 1 (éxito)</pre>	



<pre>// Instanciar MarcasModel con el mock de Query \$marcasModel = new MarcasModel(); // Llamar al método modificarMarca y almacenar el resultado \$result = \$marcasModel->modificarMarca(\$marcaModificada, \$idMarca); // Verificar que el resultado retornado sea "modificado" \$this->assertEquals('modificado', \$result); }</pre>	
testModificarMarcaError	Verifica que el método modificarMarca retorna "error" cuando ocurre un error al intentar actualizar una marca existente en la base de datos.
<p>Codigo:</p> <pre>public function testModificarMarcaError() { // Datos simulados para la modificación \$marcaModificada = 'Marca Modificada'; \$idMarca = 1; // Configurar el mock para el método save en la consulta de modificarMarca \$this->mockQuery->shouldReceive('save') ->with("UPDATE marcas SET marca = ? WHERE id = ?", [\$marcaModificada, \$idMarca]) ->andReturn(0); // Supongamos que devuelve 0 (error) // Instanciar MarcasModel con el mock de Query \$marcasModel = new MarcasModel(); // Llamar al método modificarMarca y almacenar el resultado \$result = \$marcasModel->modificarMarca(\$marcaModificada, \$idMarca); // Verificar que el resultado retornado sea "error" \$this->assertEquals('error', \$result); }</pre>	
testEditarMarca	Verifica que el método editarMarca retorna los datos correctos de una marca específica según su ID.
<p>Codigo:</p> <pre>public function testEditarMarca() { // Datos simulados para la edición \$idMarca = 1;</pre>	



```
$mockedData = ['id' => 1, 'marca' => 'Marca A', 'estado' => 1];

// Configurar el mock para el método select en la consulta de editarMarca
$this->mockQuery->shouldReceive('select')
    ->with("SELECT * FROM marcas WHERE id = $idMarca")
    ->andReturn($mockedData);

// Instanciar MarcasModel con el mock de Query
$marcasModel = new MarcasModel();

// Llamar al método editarMarca y almacenar el resultado
$result = $marcasModel->editarMarca($idMarca);

// Verificar que el resultado retornado sea el mismo que $mockedData
$this->assertEquals($mockedData, $result);
}
```

testAccionMarca

Verifica que el método accionMarca cambia correctamente el estado de una marca específica en la base de datos.

Codigo:

```
public function testAccionMarca()
{
    // Datos simulados para la acción de cambio de estado
    $estadoNuevo = 0;
    $idMarca = 1;

    $this->mockQuery->shouldReceive('save')
        ->with("UPDATE marcas SET estado = ? WHERE id = ?", [$estadoNuevo, $idMarca])
        ->andReturn(true); // Devolver true en lugar de 1

    // Instanciar MarcasModel con el mock de Query
    $marcasModel = new MarcasModel();

    // Llamar al método accionMarca y almacenar el resultado
    $result = $marcasModel->accionMarca($estadoNuevo, $idMarca);

    // Verificar que el resultado retornado sea verdadero (éxito)
    $this->assertTrue($result);
}
```



- Test de PricingModel.php

Titulo del Test	Descripción
testGetVehiculos	Verifica que el método getVehiculos retorne los vehículos activos correctamente, incluyendo marcas y tipos asociados.
<p>Codigo:</p> <pre>public function testGetVehiculos() { // Estado simulado \$estado = 1; // Datos simulados que se esperan retornar del método selectAll \$mockedData = [['id' => 1, 'nombre' => 'Vehículo A', 'marca' => 'Marca A', 'tipo' => 'Tipo A', 'estado' => 1], ['id' => 2, 'nombre' => 'Vehículo B', 'marca' => 'Marca B', 'tipo' => 'Tipo B', 'estado' => 1],]; // Configurar el mock para el método selectAll en la consulta de vehículos \$this->mockQuery->shouldReceive('selectAll') ->once() ->with("SELECT v.*, m.marca, t.tipo FROM vehiculos v INNER JOIN marcas m ON v.id_marca = m.id INNER JOIN tipos t ON v.id_tipo = t.id WHERE v.estado = \$estado") ->andReturn(\$mockedData); // Instanciar PricingModel y llamar al método getVehiculos \$pricingModel = new PricingModel(); \$result = \$pricingModel->getVehiculos(\$estado); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	



- Test de ReservasModel.php

Titulo del Test	Descripción
testGetReservasIdVehiculoVacio	Verifica que el método getReservas retorne todas las reservas cuando id_vehiculo está vacío.
Codigo: <pre>public function testGetReservasIdVehiculoVacio() { // SQL esperado cuando id_vehiculo está vacío \$sql = "SELECT r.*, c.nombre FROM reservas r INNER JOIN clientes c ON r.id_cliente = c.id"; // Datos simulados que se esperan retornar del método selectAll \$mockedData = [['id' => 1, 'nombre' => 'Reserva A'], ['id' => 2, 'nombre' => 'Reserva B']]; // Configurar el mock para el método selectAll en la consulta de reservas sin id_vehiculo \$this->mockQuery->shouldReceive('selectAll') ->with(\$sql) ->once() // Esperamos una sola llamada a selectAll con este SQL ->andReturn(\$mockedData); // Instanciar ReservasModel y llamar al método getReservas con id_vehiculo vacío \$reservasModel = new ReservasModel(); \$result = \$reservasModel->getReservas(''); // Verificar que el resultado retornado coincida con \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testGetReservasIdVehiculoEspecifico	Verifica que el método getReservas retorne las reservas específicas para un id_vehiculo dado.
Codigo: <pre>public function testGetReservasIdVehiculoEspecifico() { // Valor específico de id_vehiculo \$id_vehiculo = 1; // SQL esperado cuando id_vehiculo tiene un valor específico \$sql = "SELECT r.*, c.nombre FROM reservas r INNER JOIN clientes c ON r.id_cliente = c.id WHERE r.id_vehiculo = \$id_vehiculo";</pre>	



```
// Datos simulados que se esperan retornar del método selectAll
$mockedData = [
    ['id' => 1, 'nombre' => 'Reserva A']
];

// Configurar el mock para el método selectAll en la consulta de reservas con
id_vehiculo específico
$this->mockQuery->shouldReceive('selectAll')
    ->with($sql)
    ->once() // Esperamos una sola llamada a selectAll con este SQL
    ->andReturn($mockedData);

// Instanciar ReservasModel y llamar al método getReservas con id_vehiculo específico
$reservasModel = new ReservasModel();
$result = $reservasModel->getReservas($id_vehiculo);

// Verificar que el resultado retornado coincida con $mockedData
$this->assertEquals($mockedData, $result);
}
```

testRegistrarReserva

Verifica que el método registrarReserva inserte una nueva reserva correctamente y retorne el ID de la reserva.

Codigo:

```
public function testRegistrarReserva()
{
    // Datos simulados para la inserción de reserva
    $fecha = '2024-06-10';
    $fecha_devolucion = '2024-06-15';
    $cantidad = 1;
    $tipo = 'tipo';
    $monto = 100.50;
    $fecha_reserva = '2024-06-01';
    $observacion = 'Observación';
    $id_veh = 1;
    $id_cli = 1;

    // Configurar el mock para el método insertar en la inserción de reserva
    $this->mockQuery->shouldReceive('insertar')
        ->once()
        ->andReturn(1); // Simulamos que se insertó correctamente y devolvió un ID
}
```



<pre>// Instanciar ReservasModel y llamar al método registrarReserva con datos simulados \$reservasModel = new ReservasModel(); \$result = \$reservasModel->registrarReserva(\$fecha, \$fecha_devolucion, \$cantidad, \$tipo, \$monto, \$fecha_reserva, \$observacion, \$id_veh, \$id_cli); // Verificar que el ID retornado sea 1 \$this->assertEquals(1, \$result); }</pre>	
testRegistrarReserva_Fallo	Verifica que el método registrarReserva retorne 0 cuando falla la inserción de una nueva reserva.
<p>Codigo:</p> <pre>public function testRegistrarReserva_Fallo() { // Datos simulados para la inserción de reserva que falla \$fecha = '2024-06-10'; \$fecha_devolucion = '2024-06-15'; \$cantidad = 1; \$tipo = 'tipo'; \$monto = 100.50; \$fecha_reserva = '2024-06-01'; \$observacion = 'Observación'; \$id_veh = 1; \$id_cli = 1; // Configurar el mock para el método insertar en la inserción de reserva \$this->mockQuery->shouldReceive('insertar') ->once() ->andReturn(0); // Simulamos que la inserción falla y devuelve 0 // Instanciar ReservasModel y llamar al método registrarReserva con datos simulados \$reservasModel = new ReservasModel(); \$result = \$reservasModel->registrarReserva(\$fecha, \$fecha_devolucion, \$cantidad, \$tipo, \$monto, \$fecha_reserva, \$observacion, \$id_veh, \$id_cli); // Verificar que el resultado sea 0, indicando que la inserción falló \$this->assertEquals(0, \$result); }</pre>	
testGetVehiculos	Verifica que el método getVehiculos del ReservasModel retorne los vehículos activos correctamente.
<p>Codigo:</p> <pre>public function testGetVehiculos() { // SQL esperado en el método getVehiculos</pre>	



```
$sql = "SELECT v.*, m.marca, t.tipo FROM vehiculos v INNER JOIN marcas m ON v.id_marca = m.id INNER JOIN tipos t ON v.id_tipo = t.id WHERE v.estado != 0";

// Datos simulados que se esperan retornar del método selectAll
$mockedData = [
    ['id' => 1, 'marca' => 'Marca A', 'tipo' => 'Tipo A', 'modelo' => 'Modelo A', 'estado' => 1],
    ['id' => 2, 'marca' => 'Marca B', 'tipo' => 'Tipo B', 'modelo' => 'Modelo B', 'estado' => 1],
];

// Configurar el mock para el método selectAll en la consulta de vehículos
$this->mockQuery->shouldReceive('selectAll')
    ->with($sql)
    ->once() // Esperamos una sola llamada a selectAll con este SQL
    ->andReturn($mockedData);

// Instanciar ReservasModel y llamar al método getVehiculos
$reservasModel = new ReservasModel();
$result = $reservasModel->getVehiculos();

// Verificar que el resultado retornado coincida con $mockedData
$this->assertEquals($mockedData, $result);
}
```

testGetVehiculo

Verifica que el método getVehiculo retorne los datos correctos de un vehículo específico por su ID.

Codigo:

```
public function testGetVehiculo()
{
    $id_vehiculo = 1;

    // SQL esperado en el método getVehiculo con el ID específico
    $sql = "SELECT v.*, m.marca, t.tipo FROM vehiculos v INNER JOIN marcas m ON v.id_marca = m.id INNER JOIN tipos t ON v.id_tipo = t.id WHERE v.id = $id_vehiculo";

    // Datos simulados que se esperan retornar del método select
    $mockedData = [
        'id' => 1,
        'marca' => 'Marca A',
        'tipo' => 'Tipo A',
        'modelo' => 'Modelo A',
        'estado' => 1
    ];
}
```



```
];

// Configurar el mock para el método select en la consulta de vehículo por ID
$this->mockQuery->shouldReceive('select')
    ->with($sql)
    ->once() // Esperamos una sola llamada a select con este SQL
    ->andReturn($mockedData);

// Instanciar ReservasModel y llamar al método getVehiculo con el ID específico
$reservasModel = new ReservasModel();
$result = $reservasModel->getVehiculo($id_vehiculo);

// Verificar que el resultado retornado coincida con $mockedData
$this->assertEquals($mockedData, $result);
}
```

testGetNuevasReservas

Verifica que el método getNuevasReservas retorne las últimas cinco reservas nuevas.

Codigo:

```
public function testGetNuevasReservas()
{
    // SQL esperado en el método getNuevasReservas
    $expectedSql = "SELECT r.id, r.f_reserva, c.nombre FROM reservas r INNER JOIN clientes
c ON r.id_cliente = c.id WHERE r.estado = 0 ORDER BY r.id DESC LIMIT 5";

    // Datos simulados que se esperan retornar del método selectAll
    $mockedData = [
        ['id' => 1, 'f_reserva' => '2024-07-01', 'nombre' => 'Cliente A'],
        ['id' => 2, 'f_reserva' => '2024-06-30', 'nombre' => 'Cliente B']
    ];

    // Configurar el mock para el método selectAll en la consulta de nuevas reservas
    $this->mockQuery->shouldReceive('selectAll')
        ->with($expectedSql)
        ->once() // Esperamos una sola llamada a selectAll con este SQL
        ->andReturn($mockedData);

    // Instanciar ReservasModel y llamar al método getNuevasReservas
    $reservasModel = new ReservasModel();
    $result = $reservasModel->getNuevasReservas();
    // Verificar que el resultado retornado coincida con $mockedData
    $this->assertEquals($mockedData, $result);
}
```



testGetReserva	Verifica que el método getReserva retorne los datos correctos de una reserva específica por su ID.
<p>Codigo:</p> <pre>public function testGetReserva() { \$id_reserva = 1; // SQL esperado en el método getReserva con el ID específico \$expectedSql = "SELECT r.*, c.nombre, c.correo FROM reservas r INNER JOIN clientes c ON r.id_cliente = c.id WHERE r.id = \$id_reserva"; // Datos simulados que se esperan retornar del método select \$mockedData = ['id' => 1, 'f_reserva' => '2024-07-01', 'nombre' => 'Cliente A', 'correo' => 'clienteA@example.com']; // Configurar el mock para el método select en la consulta de reserva por ID \$this->mockQuery->shouldReceive('select') ->with(\$expectedSql) ->once() // Esperamos una sola llamada a select con este SQL ->andReturn(\$mockedData); // Instanciar ReservasModel y llamar al método getReserva con el ID específico \$reservasModel = new ReservasModel(); \$result = \$reservasModel->getReserva(\$id_reserva); // Verificar que el resultado retornado coincida con \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testActualizarEstado	Verifica que el método actualizarEstado actualice el estado de una reserva correctamente.
<p>Codigo:</p> <pre>public function testActualizarEstado() { \$id_reserva = 1; \$nuevo_estado = 1; // Nuevo estado que se va a actualizar // SQL esperado en el método actualizarEstado \$expectedSql = "UPDATE reservas SET estado = ? WHERE id = ?";</pre>	



```
// Configurar el mock para el método save en la actualización de estado
$this->mockQuery->shouldReceive('save')
    ->with($expectedSql, [$nuevo_estado, $id_reserva])
    ->once() // Esperamos una sola llamada a save con este SQL y estos datos
    ->andReturn(1); // Simulamos que la actualización fue exitosa

// Instanciar ReservasModel y llamar al método actualizarEstado con los parámetros
simulados
$reservasModel = new ReservasModel();
$result = $reservasModel->actualizarEstado($nuevo_estado, $id_reserva);

// Verificar que el resultado retornado sea 1, indicando que la actualización fue
exitosa
$this->assertEquals(1, $result);
}
```

testGetEmpresa

Verifica que el método getEmpresa retorne los datos de configuración de la empresa correctamente.

Codigo:

```
public function testGetEmpresa()
{
    // SQL esperado en el método getEmpresa
    $expectedSql = "SELECT * FROM configuracion";

    // Datos simulados que se esperan retornar del método select
    $mockedData = [
        'nombre' => 'Mi Empresa',
        'direccion' => 'Calle Principal',
        'telefono' => '123456789',
        'email' => 'info@miempresa.com'
    ];

    // Configurar el mock para el método select en la consulta de configuración de empresa
    $this->mockQuery->shouldReceive('select')
        ->with($expectedSql)
        ->once() // Esperamos una sola llamada a select con este SQL
        ->andReturn($mockedData);

    // Instanciar ReservasModel y llamar al método getEmpresa
    $reservasModel = new ReservasModel();
    $result = $reservasModel->getEmpresa();
    // Verificar que el resultado retornado coincida con $mockedData
    $this->assertEquals($mockedData, $result);
}
```



- Test de TiposModel.php

Titulo del Test	Descripción
testGetTipos	Verifica que el método getTipos retorne los tipos activos correctamente.
Codigo: <pre>public function testGetTipos() { // Estado de los tipos a buscar \$estado = 1; // Datos simulados de tipos que se esperan retornar \$mockedData = [['id' => 1, 'tipo' => 'Tipo A', 'estado' => 1], ['id' => 2, 'tipo' => 'Tipo B', 'estado' => 1],]; // Configuración de mock para el método selectAll en el modelo \$sql = "SELECT * FROM tipos WHERE estado = \$estado"; \$this->mockQuery->shouldReceive('selectAll') ->with(\$sql) ->andReturn(\$mockedData); // Instanciar TiposModel y llamar al método getTipos \$tiposModel = new TiposModel(); \$result = \$tiposModel->getTipos(\$estado); // Verificar que el resultado coincida con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRegistrarTipo	Verifica que el método registrarTipo inserte un nuevo tipo correctamente cuando no existe previamente.
Codigo: <pre>public function testRegistrarTipo() { // Tipo a registrar \$tipo = 'Nuevo Tipo'; // Configuración de mock para verificar si el tipo ya existe (caso no existente) \$this->mockQuery->shouldReceive('select') ->andReturn([]); // Simular que no existe ningún tipo con este nombre // Configuración de mock para el método save en el registro de tipo (caso exitoso)</pre>	



<pre>\$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la inserción fue exitosa // Instanciar TiposModel y llamar al método registrarTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->registrarTipo(\$tipo); // Verificar que el resultado sea "ok" (registro exitoso) \$this->assertEquals('ok', \$result); }</pre>	
testRegistrarTipoExistente	Verifica que el método registrarTipo detecte y retorne "existe" cuando el tipo ya existe.
<p>Codigo:</p> <pre>public function testRegistrarTipoExistente() { // Tipo a registrar \$tipo = 'Nuevo Tipo'; // Configuración de mock para verificar si el tipo ya existe (caso existente) \$this->mockQuery->shouldReceive('select') ->andReturn(['id' => 1, 'tipo' => 'Nuevo Tipo']); // Simular que ya existe un tipo con este nombre // Instanciar TiposModel y llamar al método registrarTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->registrarTipo(\$tipo); // Verificar que el resultado sea "existe" (tipo ya existe) \$this->assertEquals('existe', \$result); }</pre>	
testRegistrarTipoError	Verifica que el método registrarTipo retorne "error" cuando falla la inserción de un nuevo tipo.
<p>Codigo:</p> <pre>public function testRegistrarTipoError() { // Tipo a registrar \$tipo = 'Nuevo Tipo'; // Configuración de mock para verificar si el tipo ya existe (caso no existente) \$this->mockQuery->shouldReceive('select')</pre>	



<pre>->andReturn([]); // Simular que no existe ningún tipo con este nombre // Configuración de mock para el método save en el registro de tipo (caso de error) \$this->mockQuery->shouldReceive('save') ->andReturn(0); // Simular que hubo un error en la inserción // Instanciar TiposModel y llamar al método registrarTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->registrarTipo(\$tipo); // Verificar que el resultado sea "error" (error en la inserción) \$this->assertEquals('error', \$result); }</pre>	
testModificarTipo	Verifica que el método modificarTipo actualice un tipo correctamente.
<p><u>Codigo:</u></p> <pre>public function testModificarTipo() { // ID del tipo a modificar \$tipoId = 1; \$nuevoTipo = 'Tipo Modificado'; // Configuración de mock para el método save en la modificación de tipo \$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la actualización fue exitosa // Instanciar TiposModel y llamar al método modificarTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->modificarTipo(\$nuevoTipo, \$tipoId); // Verificar que el resultado sea "modificado" (actualización exitosa) \$this->assertEquals('modificado', \$result); }</pre>	
testModificarTipoError	Verifica que el método modificarTipo retorne "error" cuando falla la actualización de un tipo.
<p><u>Codigo:</u></p> <pre>public function testModificarTipoError() { // ID del tipo a modificar \$tipoId = 1; \$nuevoTipo = 'Tipo Modificado';</pre>	



```
// Configuración de mock para el método save en la modificación de tipo (caso de error)
$this->mockQuery->shouldReceive('save')
    ->andReturn(0); // Simular que hubo un error en la actualización

// Instanciar TiposModel y llamar al método modificarTipo
$tiposModel = new TiposModel();
$result = $tiposModel->modificarTipo($nuevoTipo, $tipoId);

// Verificar que el resultado sea "error" (error en la actualización)
$this->assertEquals('error', $result);
}
```

testEditarTipo

Verifica que el método editarTipo retorne los datos correctos de un tipo específico por su ID.

Codigo:

```
public function testEditarTipo()
{
    // ID del tipo a editar
    $tipoId = 1;

    // Datos simulados del tipo que se espera retornar
    $mockedTipo = [
        'id' => 1,
        'tipo' => 'Tipo A',
        'estado' => 1,
    ];

    // Configuración de mock para el método select en el modelo
    $sql = "SELECT * FROM tipos WHERE id = $tipoId";
    $this->mockQuery->shouldReceive('select')
        ->with($sql)
        ->andReturn($mockedTipo);

    // Instanciar TiposModel y llamar al método editarTipo
    $tiposModel = new TiposModel();
    $result = $tiposModel->editarTipo($tipoId);

    // Verificar que el resultado coincida con los datos esperados
    $this->assertEquals($mockedTipo, $result);
}
```




testEditarTipoNoExistente	Verifica que el método editarTipo retorne null cuando el tipo especificado no existe.
<p>Codigo:</p> <pre>public function testEditarTipoNoExistente() { // ID del tipo a editar \$tipoId = 999; // ID que no existe // Configuración de mock para el método select en el modelo (caso no existente) \$sql = "SELECT * FROM tipos WHERE id = \$tipoId"; \$this->mockQuery->shouldReceive('select') ->with(\$sql) ->andReturn(null); // Simular que no se encontró ningún tipo con ese ID // Instanciar TiposModel y llamar al método editarTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->editarTipo(\$tipoId); // Verificar que el resultado sea null (tipo no encontrado) \$this->assertNull(\$result); }</pre>	
testAccionTipo	Verifica que el método accionTipo actualice el estado de un tipo correctamente.
<p>Codigo:</p> <pre>public function testAccionTipo() { // ID del tipo a modificar \$tipoId = 1; \$nuevoEstado = 0; // Configuración de mock para el método save en la acción de tipo \$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la actualización fue exitosa // Instanciar TiposModel y llamar al método accionTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->accionTipo(\$nuevoEstado, \$tipoId); // Verificar que el resultado sea 1 (éxito en la acción) \$this->assertEquals(1, \$result); }</pre>	



testAccionTipoError	Verifica que el método accionTipo retorne 0 cuando falla la actualización del estado de un tipo.
Codigo: <pre>public function testAccionTipoError() { // ID del tipo a modificar \$tipoId = 1; \$nuevoEstado = 0; // Configuración de mock para el método save en la acción de tipo (caso de error) \$this->mockQuery->shouldReceive('save') ->andReturn(0); // Simular que hubo un error en la actualización // Instanciar TiposModel y llamar al método accionTipo \$tiposModel = new TiposModel(); \$result = \$tiposModel->accionTipo(\$nuevoEstado, \$tipoId); // Verificar que el resultado sea 0 (error en la acción) \$this->assertEquals(0, \$result); }</pre>	

- Test de UsuariosModel.php

Titulo del Test	Descripción
testGetUsuario	Verifica que el método getUsuario retorne los datos del usuario correctamente al proporcionar el usuario y la clave.
Codigo: <pre>public function testGetUsuario() { // Preparar los datos esperados \$usuario = 'admin'; \$clave = 'password'; \$mockedData = [['id' => 1, 'usuario' => 'admin', 'nombre' => 'Administrador', 'correo' => 'admin@example.com', 'estado' => 1]]; // Configurar el comportamiento esperado del mock de Query para el método select \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM usuarios WHERE usuario = '\$usuario' AND clave = '\$clave'") ->once() ->andReturn(\$mockedData);</pre>	



<pre>// Instanciar el modelo y llamar al método que queremos probar \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->getUsuario(\$usuario, \$clave); // Asegurar que el resultado coincide con los datos esperados \$this->assertEquals(\$mockedData, \$result); }</pre>	
testRegistrarUsuarioExitoso	Verifica que el método registrarUsuario inserte un nuevo usuario correctamente cuando no existe previamente.
<p>Codigo:</p> <pre>public function testRegistrarUsuarioExitoso() { // Datos simulados para el registro de un nuevo usuario \$usuario = 'nuevo_usuario'; \$nombre = 'Nuevo Usuario'; \$correo = 'nuevo_usuario@example.com'; \$telefono = '123456789'; \$clave = 'password'; // Configuración de mock para verificar si el usuario ya existe \$this->mockQuery->shouldReceive('select') ->andReturn([]); // Simular que no existe ningún usuario con este nombre o correo // Configuración de mock para el método save en el registro de usuario \$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la inserción fue exitosa // Instanciar UsuariosModel y llamar al método registrarUsuario \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->registrarUsuario(\$usuario, \$nombre, \$correo, \$telefono, \$clave); // Verificar que el resultado sea "ok" (registro exitoso) \$this->assertEquals('ok', \$result); }</pre>	
testRegistrarUsuarioExistente	Verifica que el método registrarUsuario detecte y retorne "existe" cuando el usuario ya existe.
<p>Codigo:</p> <pre>public function testRegistrarUsuarioExistente() { // Datos simulados para el registro de un nuevo usuario</pre>	



```
$usuario = 'usuario_existente';
$nombre = 'Usuario Existente';
$correo = 'existente@example.com';
$telefono = '987654321';
$clave = 'password';

// Configuración de mock para verificar si el usuario ya existe
$this->mockQuery->shouldReceive('select')
    ->andReturn([[ 'id' => 1 ]]); // Simular que el usuario ya existe en la base de
datos

// Instanciar UsuariosModel y llamar al método registrarUsuario
$usuariosModel = new UsuariosModel();
$result = $usuariosModel->registrarUsuario($usuario, $nombre, $correo, $telefono,
$clave);

// Verificar que el resultado sea "existe" (usuario ya registrado)
$this->assertEquals('existe', $result);
}
```

testRegistrarUsuarioError

Verifica que el método registrarUsuario retorne "error" cuando falla la inserción de un nuevo usuario.

Codigo:

```
public function testRegistrarUsuarioError()
{
    // Datos simulados para el registro de un nuevo usuario
    $usuario = 'nuevo_usuario';
    $nombre = 'Nuevo Usuario';
    $correo = 'nuevo_usuario@example.com';
    $telefono = '123456789';
    $clave = 'password';

    // Configuración de mock para verificar si el usuario ya existe
    $this->mockQuery->shouldReceive('select')
        ->andReturn([]); // Simular que no existe ningún usuario con este nombre o correo

    // Configuración de mock para el método save en el registro de usuario
    $this->mockQuery->shouldReceive('save')
        ->andReturn(0); // Simular que la inserción falló

    // Instanciar UsuariosModel y llamar al método registrarUsuario
    $usuariosModel = new UsuariosModel();
```



<pre> \$result = \$usuariosModel->registrarUsuario(\$usuario, \$nombre, \$correo, \$telefono, \$clave); // Verificar que el resultado sea "error" (registro fallido) \$this->assertEquals('error', \$result); } </pre>	
testGetUsuarios	Verifica que el método getUsers retorne los usuarios activos correctamente.
<p>Codigo:</p> <pre>public function testGetUsuarios() { // Estado del usuario a buscar \$estado = 1; // Datos simulados de usuarios que se esperan retornar \$mockedData = [['id' => 1, 'usuario' => 'user1', 'nombre' => 'Usuario 1', 'correo' => 'user1@example.com', 'estado' => 1], ['id' => 2, 'usuario' => 'user2', 'nombre' => 'Usuario 2', 'correo' => 'user2@example.com', 'estado' => 1],]; // Configuración de mock para el método selectAll en el modelo \$sql = "SELECT id, usuario, nombre, correo, estado FROM usuarios WHERE estado = \$estado"; \$this->mockQuery->shouldReceive('selectAll') ->with(\$sql) ->andReturn(\$mockedData); // Instanciar UsuariosModel y llamar al método getUsers \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->getUsuarios(\$estado); // Verificar que el resultado coincida con los datos esperados \$this->assertEquals(\$mockedData, \$result); } </pre>	
testModificarUsuarioExitoso	Verifica que el método modificarUsuario actualice un usuario correctamente.
<p>Codigo:</p> <pre>public function testModificarUsuarioExitoso() { // Datos simulados para modificar un usuario existente </pre>	



```
$usuario = 'usuario_modificado';
$nombre = 'Usuario Modificado';
$correo = 'modificado@example.com';
$telefono = '987654321';
$idUsuario = 1;

// Configuración de mock para el método save en la modificación de usuario
$this->mockQuery->shouldReceive('save')
    ->andReturn(1); // Simular que la actualización fue exitosa

// Instanciar UsuariosModel y llamar al método modificarUsuario
$usuariosModel = new UsuariosModel();
$result = $usuariosModel->modificarUsuario($usuario, $nombre, $correo, $telefono,
$idUsuario);

// Verificar que el resultado sea "modificado" (actualización exitosa)
$this->assertEquals('modificado', $result);
}
```

testModificarUsuarioFallido

Verifica que el método modificarUsuario retorne "error" cuando falla la actualización de un usuario.

Codigo:

```
public function testModificarUsuarioFallido()
{
    // Datos simulados para modificar un usuario existente
    $usuario = 'usuario_modificado';
    $nombre = 'Usuario Modificado';
    $correo = 'modificado@example.com';
    $telefono = '987654321';
    $idUsuario = 999; // ID inexistente para forzar un fallo

    // Configuración de mock para el método save en la modificación de usuario
    $this->mockQuery->shouldReceive('save')
        ->andReturn(0); // Simular que la actualización falló

    // Instanciar UsuariosModel y llamar al método modificarUsuario
    $usuariosModel = new UsuariosModel();
    $result = $usuariosModel->modificarUsuario($usuario, $nombre, $correo, $telefono,
$idUsuario);

    // Verificar que el resultado sea "error" (actualización fallida)
    $this->assertEquals('error', $result);
}
```



testEditarUser	Verifica que el método editarUser retorne los datos correctos de un usuario específico por su ID.
<p>Codigo:</p> <pre>public function testEditarUser() { // ID del usuario a editar \$userId = 1; // Datos simulados del usuario que se espera retornar \$mockedUser = ['id' => 1, 'usuario' => 'user1', 'nombre' => 'Usuario 1', 'correo' => 'user1@example.com', 'telefono' => '123456789', 'estado' => 1,]; // Configuración de mock para el método select en el modelo \$sql = "SELECT * FROM usuarios WHERE id = \$userId"; \$this->mockQuery->shouldReceive('select') ->with(\$sql) ->andReturn(\$mockedUser); // Instanciar UsuariosModel y llamar al método editarUser \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->editarUser(\$userId); // Verificar que el resultado coincida con los datos esperados \$this->assertEquals(\$mockedUser, \$result); }</pre>	
testAccionUser	Verifica que el método accionUser actualice el estado de un usuario correctamente.
<p>Codigo:</p> <pre>public function testAccionUser() { // Estado y ID del usuario para cambiar \$estado = 0; // Estado inactivo \$idUserio = 1; // Configuración de mock para el método save en la acción de usuario \$this->mockQuery->shouldReceive('save')</pre>	



<pre>->andReturn(1); // Simular que la actualización fue exitosa // Instanciar UsuariosModel y llamar al método accionUser \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->accionUser(\$estado, \$idUserio); // Verificar que el resultado sea 1 (indicando éxito en la actualización) \$this->assertEquals(1, \$result); }</pre>	
testModificarPass	Verifica que el método modificarPass actualice la contraseña de un usuario correctamente.
<p><u>Codigo:</u></p> <pre>public function testModificarPass() { // Nueva contraseña y ID del usuario \$nuevaClave = 'nueva_password'; \$idUserio = 1; // Configuración de mock para el método save en la modificación de contraseña \$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la actualización fue exitosa // Instanciar UsuariosModel y llamar al método modificarPass \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->modificarPass(\$nuevaClave, \$idUserio); // Verificar que el resultado sea 1 (indicando éxito en la actualización) \$this->assertEquals(1, \$result); }</pre>	
testGetEmpresa	Verifica que el método getEmpresa retorne los datos de configuración de la empresa correctamente.
<p><u>Codigo:</u></p> <pre>public function testGetEmpresa() { // Datos simulados que se esperan retornar del método select \$mockedData = [['id' => 1, 'nombre' => 'Empresa A', 'direccion' => 'Calle Principal', 'telefono' => '123456789']]; // Configurar el mock para el método select en la consulta de empresa \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM configuracion")</pre>	



<pre>->andReturn(\$mockedData); // Instanciar UsuariosModel y llamar al método getEmpresa \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->getEmpresa(); // Verificar que el resultado retornado sea el mismo que \$mockedData \$this->assertEquals(\$mockedData, \$result); }</pre>	
testModificarDato	Verifica que el método modificarDato actualice los datos de un usuario correctamente.
<p>Codigo:</p> <pre>public function testModificarDato() { // Datos simulados para modificar los datos de un usuario existente \$usuario = 'usuario_modificado'; \$nombre = 'Usuario Modificado'; \$apellido = 'Apellido Modificado'; \$correo = 'modificado@example.com'; \$telefono = '987654321'; \$direccion = 'Calle Modificada'; \$perfil = 'imagen_modificada.jpg'; \$idUserio = 1; // Configuración de mock para el método save en la modificación de datos de usuario \$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la actualización fue exitosa // Instanciar UsuariosModel y llamar al método modificarDato \$usuariosModel = new UsuariosModel(); \$result = \$usuariosModel->modificarDato(\$usuario, \$nombre, \$apellido, \$correo, \$telefono, \$direccion, \$perfil, \$idUserio); // Verificar que el resultado sea 1 (indicando éxito en la actualización) \$this->assertEquals(1, \$result); }</pre>	
testModificarDatoFallido	Verifica que el método modificarDato retorne 0 cuando falla la actualización de los datos de un usuario.
<p>Codigo:</p> <pre>public function testModificarDatoFallido() { // Datos simulados para modificar los datos de un usuario existente</pre>	



```
$usuario = 'usuario_modificado';
$nombre = 'Usuario Modificado';
$apellido = 'Apellido Modificado';
$correo = 'modificado@example.com';
$telefono = '987654321';
$direccion = 'Calle Modificada';
$perfil = 'imagen_modificada.jpg';
$idUsuario = 999; // ID inexistente para forzar un fallo

// Configuración de mock para el método save en la modificación de datos de usuario
$this->mockQuery->shouldReceive('save')
    ->andReturn(0); // Simular que la actualización falló

// Instanciar UsuariosModel y llamar al método modificarDato
$usuariosModel = new UsuariosModel();
$result = $usuariosModel->modificarDato($usuario, $nombre, $apellido, $correo,
    $telefono, $direccion, $perfil, $idUsuario);

// Verificar que el resultado sea 0 (indicando fallo en la actualización)
$this->assertEquals(0, $result);
}
```

- Test de VehiculosModel.php

Titulo del Test	Descripción
testGetDatos	Verifica que el método getDatos de VehiculosModel devuelve correctamente los datos de una tabla específica.
Codigo: <pre>public function testGetDatos() { // Configurar el mock para simular la llamada a selectAll con la consulta correcta \$this->mockQuery->shouldReceive('selectAll') ->with('SELECT * FROM tabla WHERE estado = 1') ->andReturn([['id' => 1, 'campo1' => 'valor1', 'estado' => 1], ['id' => 2, 'campo1' => 'valor2', 'estado' => 1],]); \$vehiculosModel = new VehiculosModel(); // Llamar al método y almacenar el resultado \$result = \$vehiculosModel->getDatos('tabla'); // Verificar que el resultado obtenido es el esperado</pre>	



```
$this->assertEquals([
    ['id' => 1, 'campo1' => 'valor1', 'estado' => 1],
    ['id' => 2, 'campo1' => 'valor2', 'estado' => 1],
], $result);
}
```

testGetVehiculos

Verifica que el método getVehiculos de VehiculosModel devuelve correctamente los vehículos con un estado específico.

Codigo:

```
public function testGetVehiculos()
{
    // Estado de vehículos a buscar
    $estado = 1;

    // Configurar el mock para simular la llamada a selectAll con la consulta correcta
    $this->mockQuery->shouldReceive('selectAll')
        ->with("SELECT v.*, m.marca, t.tipo FROM vehiculos v INNER JOIN
marcas m ON v.id_marca = m.id INNER JOIN tipos t ON v.id_tipo = t.id WHERE v.estado =
$estado")
        ->andReturn([
            ['id' => 1, 'placa' => 'ABC123', 'marca' => 'Marca A', 'tipo' =>
'Tipo A', 'estado' => 1],
            ['id' => 2, 'placa' => 'XYZ789', 'marca' => 'Marca B', 'tipo' =>
'Tipo B', 'estado' => 1],
        ]);

    $vehiculosModel = new VehiculosModel();

    // Llamar al método y almacenar el resultado
    $result = $vehiculosModel->getVehiculos($estado);

    // Verificar que el resultado obtenido es el esperado
    $this->assertEquals([
        ['id' => 1, 'placa' => 'ABC123', 'marca' => 'Marca A', 'tipo' => 'Tipo A',
'estado' => 1],
        ['id' => 2, 'placa' => 'XYZ789', 'marca' => 'Marca B', 'tipo' => 'Tipo B',
'estado' => 1],
    ], $result);
}
```



testVehiculos	Verifica que el método vehiculos de VehiculosModel devuelve correctamente los vehículos con estado 1 o 2.
<p>Codigo:</p> <pre>public function testVehiculos() { // Configurar el mock para simular la llamada a selectAll con la consulta correcta \$this->mockQuery->shouldReceive('selectAll') ->with("SELECT v.*, m.marca, t.tipo FROM vehiculos v INNER JOIN marcas m ON v.id_marca = m.id INNER JOIN tipos t ON v.id_tipo = t.id WHERE v.estado = 1 OR v.estado = 2") ->andReturn([['id' => 1, 'placa' => 'ABC123', 'marca' => 'Marca A', 'tipo' => 'Tipo A', 'estado' => 1], ['id' => 2, 'placa' => 'XYZ789', 'marca' => 'Marca B', 'tipo' => 'Tipo B', 'estado' => 2],]); \$vehiculosModel = new VehiculosModel(); // Llamar al método y almacenar el resultado \$result = \$vehiculosModel->vehiculos(); // Verificar que el resultado obtenido es el esperado \$this->assertEquals([['id' => 1, 'placa' => 'ABC123', 'marca' => 'Marca A', 'tipo' => 'Tipo A', 'estado' => 1], ['id' => 2, 'placa' => 'XYZ789', 'marca' => 'Marca B', 'tipo' => 'Tipo B', 'estado' => 2],], \$result); }</pre>	
testRegistrarVehiculo	Verifica que el método registrarVehiculo de VehiculosModel registra correctamente un vehículo nuevo y devuelve "ok".
<p>Codigo:</p> <pre>public function testRegistrarVehiculo() { // Placa de vehículo que no existe \$placaNueva = 'XYZ789'; // Configurar el mock para que select devuelva que el vehículo no existe \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM vehiculos WHERE placa = '\$placaNueva'")</pre>	



```
->andReturn([]); // Simular que el vehículo no existe

// Configurar el mock para el método save en el registro de vehículo (caso exitoso)
$this->mockQuery->shouldReceive('save')
    ->andReturn(1); // Simular que la inserción fue exitosa

$vehiculosModel = new VehiculosModel();

// Llamar al método y almacenar el resultado
$result = $vehiculosModel->registrarVehiculo(
    $placaNueva, 10, 100, 1000, 'Modelo B', 60000, 'Manual', 5, 'Mediano', 'Gasolina',
    'imagen.jpg', 2, 2
);
// Verificar que el resultado retornado sea "ok"
$this->assertEquals('ok', $result);
}
```

testRegistrarVehiculoExistente

Verifica que el método registrarVehiculo de VehiculosModel devuelve "existe" cuando se intenta registrar un vehículo ya existente.

Codigo:

```
public function testRegistrarVehiculoExistente()
{
    // Placa de vehículo que ya existe
    $placaExistente = 'ABC123';

    // Configurar el mock para que select devuelva un vehículo existente
    $this->mockQuery->shouldReceive('select')
        ->with("SELECT * FROM vehiculos WHERE placa = '$placaExistente'")
        ->andReturn(['id' => 1, 'placa' => $placaExistente]); // Simular que
    el vehículo ya existe

    $vehiculosModel = new VehiculosModel();

    // Llamar al método y almacenar el resultado
    $result = $vehiculosModel->registrarVehiculo(
        $placaExistente, 10, 100, 1000, 'Modelo A', 50000, 'Automatica', 5, 'Grande',
        'Gasolina', 'imagen.jpg', 1, 1
    );

    // Verificar que el resultado retornado sea "existe"
    $this->assertEquals('existe', $result);
}
```



testRegistrarVehiculoError	Verifica que el método registrarVehiculo de VehiculosModel devuelve "error" cuando falla el registro de un vehículo nuevo.
<p>Codigo:</p> <pre>public function testRegistrarVehiculoError() { // Placa de vehículo que no existe \$placaNueva = 'XYZ789'; // Configurar el mock para que select devuelva un vehículo que no existe \$this->mockQuery->shouldReceive('select') ->with("SELECT * FROM vehiculos WHERE placa = '\$placaNueva'") ->andReturn([]); // Simular que el vehículo no existe // Configurar el mock para que save devuelva 0, indicando error en la inserción \$this->mockQuery->shouldReceive('save') ->andReturn(0); // Simular que la inserción falla \$vehiculosModel = new VehiculosModel(); // Llamar al método y almacenar el resultado \$result = \$vehiculosModel->registrarVehiculo(\$placaNueva, 10, 100, 1000, 'Modelo B', 60000, 'Manual', 4, 'Mediano', 'Gasolina', 'imagen.jpg', 2, 2); // Verificar que el resultado retornado sea "error" \$this->assertEquals('error', \$result); }</pre>	
testModificarVehiculo	Verifica que el método modificarVehiculo de VehiculosModel actualiza correctamente los datos de un vehículo y devuelve "modificado".
<p>Codigo:</p> <pre>public function testModificarVehiculo() { \$placa = 'ABC123'; \$precio_hora = 15; \$precio_dia = 150; \$precio_mes = 1500; \$modelo = 'Modelo Actualizado'; \$kilometraje = 60000; \$transmision = 'Manual'; \$asientos = 4;</pre>	



```
$equipaje = 'Mediano';
$combustible = 'Gasolina';
$imgNombre = 'imagen_actualizada.jpg';
$tipo = 2;
$marca = 2;
$id = 1;
// Configurar mock para el método save en la modificación de vehículo
$this->mockQuery->shouldReceive('save')
    ->andReturn(1); // Simular que la actualización fue exitosa

$vehiculosModel = new VehiculosModel();
// Llamar al método y almacenar el resultado
$result = $vehiculosModel->modificarVehiculo(
    $placa, $precio_hora, $precio_dia, $precio_mes, $modelo, $kilometraje,
    $transmision,
    $asientos, $equipaje, $combustible, $imgNombre, $tipo, $marca, $id
);
// Verificar que el resultado retornado sea "modificado"
$this->assertEquals('modificado', $result);
}
```

testModificarVehiculoError

Verifica que el método modificarVehiculo de VehiculosModel devuelve "error" cuando falla la actualización de un vehículo.

Codigo:

```
public function testModificarVehiculoError()
{
    // Datos del vehículo a modificar
    $placa = 'ABC123';
    $precio_hora = 10;
    $precio_dia = 100;
    $precio_mes = 1000;
    $modelo = 'Modelo A';
    $kilometraje = 50000;
    $transmision = 'Automatica';
    $asientos = 5;
    $equipaje = 'Grande';
    $combustible = 'Gasolina';
    $imgNombre = 'imagen.jpg';
    $tipo = 1;
    $marca = 1;
    $id = 1;

    // Configurar el mock para simular la llamada a save con el SQL y los datos
}
```



```
$this->mockQuery->shouldReceive('save')
    ->andReturn(0); // Simular que la modificación falla y no se realiza

$vehiculosModel = new VehiculosModel();

// Llamar al método y almacenar el resultado
$result = $vehiculosModel->modificarVehiculo(
    $placa, $precio_hora, $precio_dia, $precio_mes,
    $modelo, $kilometraje, $transmision, $asientos, $equipaje,
    $combustible, $imgNombre, $tipo, $marca, $id
);
// Verificar que el resultado retornado es "error"
$this->assertEquals('error', $result);
}
```

testEditarVehiculo

Verifica que el método editarVeh de VehiculosModel devuelve correctamente los datos de un vehículo específico.

Codigo:

```
public function testEditarVehiculo()
{
    $idVehiculo = 1;

    // Datos simulados del vehículo que se espera retornar
    $mockedVehiculo = [
        'id' => $idVehiculo,
        'placa' => 'ABC123',
        'precio_hora' => 10,
        'precio_dia' => 100,
        'precio_mes' => 1000,
        'modelo' => 'Modelo A',
        'kilometraje' => 50000,
        'transmision' => 'Automatica',
        'asientos' => 5,
        'equipaje' => 'Grande',
        'combustible' => 'Gasolina',
        'foto' => 'imagen.jpg',
        'id_tipo' => 1,
        'id_marca' => 1,
        'estado' => 1,
    ];

    // Configuración de mock para el método select en el modelo
    $sql = "SELECT * FROM vehiculos WHERE id = $idVehiculo";
```




<pre>\$this->mockQuery->shouldReceive('select') ->with(\$sql) ->andReturn(\$mockedVehiculo); \$vehiculosModel = new VehiculosModel(); // Llamar al método y almacenar el resultado \$result = \$vehiculosModel->editarVeh(\$idVehiculo); // Verificar que el resultado coincida con los datos esperados \$this->assertEquals(\$mockedVehiculo, \$result); }</pre>	
testAccionVehiculo	Verifica que el método accionVeh de VehiculosModel actualiza correctamente el estado de un vehículo y devuelve 1.
Codigo: <pre>public function testAccionVehiculo() { \$estado = 0; \$idVehiculo = 1; // Configurar mock para el método save en la acción de vehículo \$this->mockQuery->shouldReceive('save') ->andReturn(1); // Simular que la actualización fue exitosa \$vehiculosModel = new VehiculosModel(); // Llamar al método y almacenar el resultado \$result = \$vehiculosModel->accionVeh(\$estado, \$idVehiculo); // Verificar que el resultado sea 1 (éxito en la acción) \$this->assertEquals(1, \$result); }</pre>	
testBuscarVehiculo	Verifica que el método buscarVehiculo de VehiculosModel devuelve correctamente los vehículos que coinciden con un valor buscado.
Codigo: <pre>public function testBuscarVehiculo() { \$valor = 'ABC'; // Datos simulados de vehículos que se esperan retornar</pre>	



```
$mockedVehiculos = [  
    ['id' => 1, 'placa' => 'ABC123', 'tipo' => 'SUV', 'marca' => 'Toyota', 'estado'  
=> 1],  
    ['id' => 2, 'placa' => 'XYZ789', 'tipo' => 'Sedan', 'marca' => 'Honda', 'estado'  
=> 1],  
];  
  
// Configuración del mock para el método selectAll en el modelo  
$this->mockQuery->shouldReceive('selectAll')  
    ->once() // Asegura que se llame exactamente una vez  
    ->andReturnUsing(function ($sql) use ($valor, $mockedVehiculos) {  
        // Verificar que la consulta contenga el valor buscado  
        if (strpos($sql, $valor) !== false) {  
            return $mockedVehiculos;  
        }  
        return [];  
    });  
  
// Instanciamos el modelo de vehículos  
$vehiculosModel = new VehiculosModel();  
  
// Llamamos al método buscarVehiculo y almacenamos el resultado  
$result = $vehiculosModel->buscarVehiculo($valor);  
  
// Verificamos que el resultado coincida con los datos esperados  
$this->assertEquals($mockedVehiculos, $result);  
}
```

7. Reporte de Pruebas guiadas por el comportamiento (BDD Given When Then)

El enfoque de pruebas guiadas por el comportamiento (Behavior Driven Development, BDD) se basa en describir el comportamiento esperado de un sistema utilizando un lenguaje común que sea comprensible tanto para los desarrolladores como para los stakeholders no técnicos. Una de las estructuras más comunes en BDD es el formato Given-When-Then:

Given (Dado)	Describe el contexto inicial del escenario de prueba.
When (Cuando)	Describe el evento o acción que se ejecuta.
Then (Entonces)	Describe el resultado esperado después de la acción.

Este formato ayuda a establecer criterios de aceptación claros y facilita la comunicación entre los diferentes miembros del equipo. Los reportes generados mediante esta metodología permiten una mejor comprensión de los requisitos y su correcta implementación.

- Instalación de Pruebas Behat

El comando **composer require --dev behat/behata** indica que se está instalando la biblioteca de pruebas Behat para un proyecto PHP, con la opción **--dev** que la configura como una dependencia de desarrollo.

```
Windows PowerShell
Instale la versión más reciente de PowerShell para obtener nuevas caracter
PS C:\Users\JOSE\Desktop\Proyecto: composer require --dev behat/behata
./composer.json has been updated
Running composer update behat/behata
Loading composer repositories with package information
Updating dependencies
Lock file operations: 22 installs, 0 updates, 0 removals
- Locking behat/behata (v3.14.0)
- Locking behat/gherkin (v4.9.0)
- Locking behat/transliterator (v1.5.0)
- Locking psr/container (2.0.2)
- Locking psr/event-dispatcher (1.0.0)
- Locking symfony/config (v7.1.1)
- Locking symfony/console (v7.1.2)
- Locking symfony/dependency-injection (v7.1.2)
- Locking symfony/deprecation-contracts (v3.5.0)
- Locking symfony/event-dispatcher (v7.1.1)
- Locking symfony/event-dispatcher-contracts (v3.5.0)
- Locking symfony/filesystem (v7.1.2)
- Locking symfony/polyfill-ctype (v1.30.0)
- Locking symfony/polyfill-intl-grapheme (v1.30.0)
- Locking symfony/polyfill-intl-normalizer (v1.30.0)
- Locking symfony/polyfill-mbstring (v1.30.0)
- Locking symfony/service-contracts (v3.5.0)
- Locking symfony/string (v7.1.2)
- Locking symfony/translation (v7.1.1)
- Locking symfony/translation-contracts (v3.5.0)
- Locking symfony/var-exporter (v7.1.2)
- Locking symfony/yaml (v7.1.1)
```

- Instalar las dependencias para el Proyecto

Se está ejecutando el comando **composer require** para instalar las dependencias de un proyecto de desarrollo. Este comando especifica las dependencias que se necesitan para el proyecto, que en este caso son las librerías **behat/behat**, **behat/mink** y **behat/mink-extension**.

```
Windows PowerShell
PS C:\Users\JOSE\Desktop\Proyecto> code .
PS C:\Users\JOSE\Desktop\Proyecto> composer require --dev behat/behat behat/mink behat/mink-extension
./composer.json has been updated
Running composer update behat/behat behat/mink behat/mink-extension
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
- Locking behat/mink (v1.11.0)
- Locking behat/mink-extension (v1.3.3)
- Locking symfony/css-selector (v7.1.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
- Downloading symfony/css-selector (v7.1.1)
- Downloading behat/mink (v1.11.0)
- Downloading behat/mink-extension (v1.3.3)
- Installing symfony/css-selector (v7.1.1): Extracting archive
- Installing behat/mink (v1.11.0): Extracting archive
- Installing behat/mink-extension (v1.3.3): Extracting archive
4 package suggestions were added by new dependencies, use 'composer suggest' to see details.
Package behat/mink-extension is abandoned, you should avoid using it. Use friends-of-behat/mink-extension
Package symfony/debug is abandoned, you should avoid using it. Use symfony/error-handler instead.
Generating autoload files
27 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
No security vulnerability advisories found.
Using version ^2.5 for behat/behat
Using version ^1.11 for behat/mink
Using version ^1.3 for behat/mink-extension
PS C:\Users\JOSE\Desktop\Proyecto>
```

- Se esta Actualizando y instalando “Symfony”

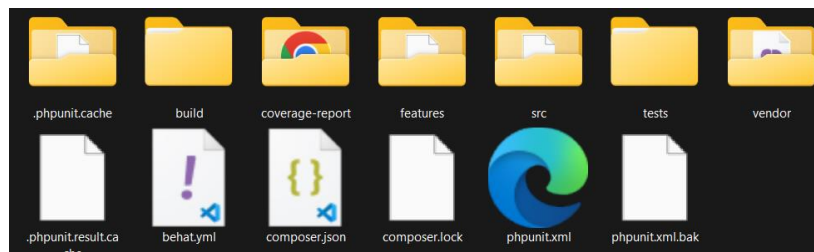
Se observa que la mayoría de los comandos están relacionados con la librería Symfony, una popular herramienta para el desarrollo web con PHP, con comandos como "Upgrading symfony/yaml" o "Installing symfony/translation-contracts", lo que sugiere que se está actualizando o instalando un proyecto que utiliza Symfony.

```
Windows PowerShell
- Upgrading symfony/yaml (v2.8.52 => v6.4.8): Extracting archive
- Installing symfony/translation-contracts (v2.5.3): Extracting archive
- Upgrading symfony/translation (v2.8.52 => v4.4.47): Extracting archive
- Installing psr/event-dispatcher (1.0.0): Extracting archive
- Installing symfony/event-dispatcher-contracts (v3.5.0): Extracting archive
- Upgrading symfony/event-dispatcher (v2.8.52 => v5.4.40): Extracting archive
- Installing psr/container (1.1.2): Extracting archive
- Installing symfony/service-contracts (v2.5.3): Extracting archive
- Upgrading symfony/dependency-injection (v2.8.52 => v4.4.49): Extracting archive
- Installing symfony/polyfill-intl-normalizer (v1.30.0): Extracting archive
- Installing symfony/polyfill-intl-grapheme (v1.30.0): Extracting archive
- Installing symfony/string (v6.4.9): Extracting archive
- Installing symfony/polyfill-php73 (v1.30.0): Extracting archive
- Upgrading symfony/console (v2.8.52 => v5.4.41): Extracting archive
- Installing behat/transliterator (v1.5.0): Extracting archive
- Upgrading behat/behat (v2.5.5 => v3.14.0): Extracting archive
- Upgrading behat/mink-extension (v1.3.3 => 2.3.1): Extracting archive
- Installing symfony/polyfill-php72 (v1.30.0): Extracting archive
- Installing symfony/polyfill-intl-idn (v1.30.0): Extracting archive
- Installing symfony/mime (v6.4.9): Extracting archive
- Installing symfony/http-client-contracts (v3.5.0): Extracting archive
- Upgrading psr/log (1.1.4 => 2.0.0): Extracting archive
- Installing symfony/http-client (v6.4.9): Extracting archive
- Installing masterminds/html5 (2.9.0): Extracting archive
- Installing symfony/dom-crawler (v6.4.8): Extracting archive
- Installing symfony/browser-kit (v6.4.8): Extracting archive
- Installing fabpot/goutte (v4.0.3): Extracting archive
- Installing behat/mink-browserkit-driver (v2.2.0): Extracting archive
- Installing behat/mink-goutte-driver (v2.0.0): Extracting archive
```

- Se usa el comando emuse/behat
Un comando de consola que está actualizando un proyecto de software usando Composer, una herramienta para gestionar dependencias de proyectos. El comando instala dos paquetes y luego informa que algunos paquetes que se usaban en el proyecto están desactualizados y que deberían ser reemplazados por otros.

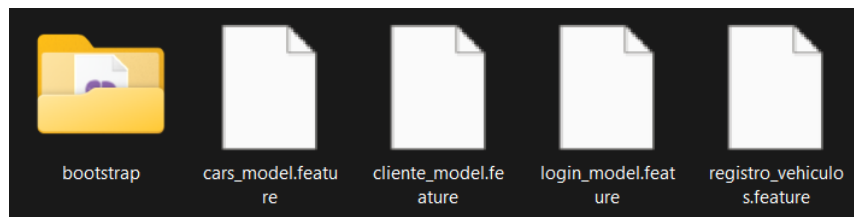
```
PS C:\Users\JOSE\Desktop\Proyecto> composer require --dev emuse/behat-html-formatter
./composer.json has been updated
Running composer update emuse/behat-html-formatter
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
- Locking emuse/behat-html-formatter (v2.0.0)
- Locking twig/twig (v3.10.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
- Downloading twig/twig (v3.10.3)
- Downloading emuse/behat-html-formatter (v2.0.0)
- Installing twig/twig (v3.10.3): Extracting archive
- Installing emuse/behat-html-formatter (v2.0.0): Extracting archive
Package behat/mink-extension is abandoned, you should avoid using it. Use friends-of-behat/mink-extension instead.
Package behat/mink-goutte-driver is abandoned, you should avoid using it. Use behat/mink-browserkit-driver instead.
Package fabpot/goutte is abandoned, you should avoid using it. Use symfony/browser-kit instead.
Generating autoload files
52 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
Using version ^2.0 for emuse/behat-html-formatter
PS C:\Users\JOSE\Desktop\Proyecto>
```

- Test Para los Escenarios de “features”

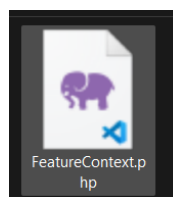


Creación de Features

- En este punto se usa el GIVE WHEN THEN



Carpeta del “bootstrap”





- Escenario para cars_model.feature

CONSULTA DE VEHÍCULOS Y DETALLES DE VEHÍCULO EN EL SISTEMA
Scenario: Consultar lista de vehículos activos
Given que tengo vehículos en el sistema con estado "activo"
When consulto la lista de vehículos
Then debería obtener una lista de vehículos con estado "activo"
Scenario: Consultar detalles de un vehículo específico
Given que tengo un vehículo con ID "1"
When consulto los detalles del vehículo con ID "1"
Then debería ver los detalles del vehículo con ID "1"

- Escenario para cliente_model.feature

REGISTRO DE CLIENTE Y VERIFICAR EL PROCESO DE REGISTRO DE UN NUEVO CLIENTE EN EL SISTEMA										
Scenario: Registro exitoso de un nuevo cliente										
Given que tengo acceso al sistema como administrador										
When registro un nuevo cliente con los siguientes datos:										
<table><tr><td> DNI</td><td> Nombre</td><td> Teléfono</td><td> Dirección</td><td> </td></tr><tr><td> 12345678A</td><td> Juan Pérez</td><td> 987654321</td><td> Calle Principal, 123</td><td> </td></tr></table>	DNI	Nombre	Teléfono	Dirección		12345678A	Juan Pérez	987654321	Calle Principal, 123	
DNI	Nombre	Teléfono	Dirección							
12345678A	Juan Pérez	987654321	Calle Principal, 123							
Then debería ver el mensaje "Cliente registrado correctamente."										
Scenario: Intento de registro de un cliente que ya existe										
Given que tengo acceso al sistema como administrador										



When intento registrar un nuevo cliente con los siguientes datos:

DNI	Nombre	Teléfono	Dirección	
12345678A	Juan Pérez	987654321	Calle Principal, 123	

Then debería ver el mensaje "Ya existe un cliente con ese nombre."

Scenario: Falla en el registro de un nuevo cliente

Given que tengo acceso al sistema como administrador

When intento registrar un nuevo cliente con datos incorrectos

Then debería ver el mensaje "Ocurrió un error al intentar registrar al cliente."

- Escenario para login_model.feature

VERIFICACIÓN DE INICIO DE SESIÓN EN EL SISTEMA
Scenario: Verificar inicio de sesión exitoso con correo electrónico en tabla de usuarios
Given que tengo un usuario con correo "john@example.com" en la tabla "usuarios"
When verifico el inicio de sesión con correo electrónico "john@example.com" y contraseña "hashed_password"
Then debería obtener un resultado exitoso de verificación de inicio de sesión
Scenario: Verificar inicio de sesión exitoso con correo electrónico en tabla de usuarios
Given que tengo un usuario con correo "admin@example.com" en la tabla "usuarios"



When verifico el inicio de sesión con correo electrónico "admin@example.com" y contraseña "hashed_password" Then debería obtener un resultado exitoso de verificación de inicio de sesión
Scenario: Verificar inicio de sesión fallido con contraseña incorrecta para usuario admin
Given que tengo un usuario con correo "admin@example.com" en la tabla "usuarios" When verifico el inicio de sesión con correo electrónico "admin@example.com" y contraseña "hashed" Then debería obtener un resultado fallido de verificación de inicio de sesión
Scenario: Verificar inicio de sesión fallido con contraseña incorrecta para usuario ana
Given que tengo un usuario con correo "ana@example.com" en la tabla "usuarios" When verifico el inicio de sesión con correo electrónico "ana@example.com" y contraseña "hashed" Then debería obtener un resultado fallido de verificación de inicio de sesión

- Escenario para registro_vehiculos.feature

VERIFICAR EL REGISTRO, MODIFICACIÓN Y BÚSQUEDA DE VEHÍCULOS EN EL SISTEMA.
Scenario: Registrar un nuevo vehículo exitosamente
When registro un nuevo vehículo con los siguientes datos: Placa Precio Hora Precio Día Precio Mes Modelo Kilometraje Transmisión Asientos Equipaje Combustible Imagen Tipo Marca ABC123 10 50 300 Modelo1 10000 Automática 5 Grande Gasolina vehiculo.jpg 1 1 Then debería ver el mensaje "Vehículo registrado correctamente."
Scenario: Intento de registrar un vehículo que ya existe



When intento registrar un nuevo vehículo con los siguientes datos:																																
<table><tr><td> Placa</td><td> Precio Hora</td><td> Precio Día</td><td> Precio Mes</td><td> Modelo</td><td> Kilometraje</td><td> Transmisión</td><td> </td></tr><tr><td>Asientos</td><td>Equipaje</td><td>Combustible</td><td>Imagen</td><td> Tipo</td><td>Marca</td><td> </td><td></td></tr><tr><td> XYZ789</td><td> 15</td><td> 60</td><td> 350</td><td> Modelo2</td><td> 8000</td><td> Manual</td><td> 4</td></tr><tr><td> Mediano</td><td> Diésel</td><td> carro.jpg</td><td> 2</td><td> 2</td><td> </td><td></td><td></td></tr></table>	Placa	Precio Hora	Precio Día	Precio Mes	Modelo	Kilometraje	Transmisión		Asientos	Equipaje	Combustible	Imagen	Tipo	Marca			XYZ789	15	60	350	Modelo2	8000	Manual	4	Mediano	Diésel	carro.jpg	2	2			
Placa	Precio Hora	Precio Día	Precio Mes	Modelo	Kilometraje	Transmisión																										
Asientos	Equipaje	Combustible	Imagen	Tipo	Marca																											
XYZ789	15	60	350	Modelo2	8000	Manual	4																									
Mediano	Diésel	carro.jpg	2	2																												
Then debería ver el mensaje "Ya existe un vehículo con esa placa."																																
Scenario: Falla en el registro de un nuevo vehículo																																
When intento registrar un nuevo vehículo con datos incorrectos																																
Then debería ver el mensaje "Ocurrió un error al intentar registrar el vehículo."																																
Scenario: Modificar un vehículo existente																																
Given existe un vehículo con id 1																																
When modifiko el vehículo con id 1 con los siguientes datos:																																
<table><tr><td> Placa</td><td> Precio Hora</td><td> Precio Día</td><td> Precio Mes</td><td> Modelo</td><td> Kilometraje</td><td> Transmisión</td><td> </td></tr><tr><td>Asientos</td><td>Equipaje</td><td>Combustible</td><td>Imagen</td><td> Tipo</td><td>Marca</td><td> </td><td></td></tr><tr><td> ABC123</td><td> 12</td><td> 55</td><td> 320</td><td> Modelo1</td><td> 12000</td><td> Automática</td><td> 5</td></tr><tr><td> Grande</td><td> Gasolina</td><td> vehiculo.jpg</td><td> 1</td><td> 1</td><td> </td><td></td><td></td></tr></table>	Placa	Precio Hora	Precio Día	Precio Mes	Modelo	Kilometraje	Transmisión		Asientos	Equipaje	Combustible	Imagen	Tipo	Marca			ABC123	12	55	320	Modelo1	12000	Automática	5	Grande	Gasolina	vehiculo.jpg	1	1			
Placa	Precio Hora	Precio Día	Precio Mes	Modelo	Kilometraje	Transmisión																										
Asientos	Equipaje	Combustible	Imagen	Tipo	Marca																											
ABC123	12	55	320	Modelo1	12000	Automática	5																									
Grande	Gasolina	vehiculo.jpg	1	1																												
Then debería ver el mensaje "Vehículo modificado correctamente."																																
Scenario: Falla al intentar modificar un vehículo																																
Given existe un vehículo con id 2																																
When intento modificar el vehículo con id 2 con datos incorrectos																																
Then debería ver el mensaje "Ocurrió un error al intentar modificar el vehículo."																																
Scenario: Buscar vehículo por placa, tipo o marca																																
Given existen vehículos registrados en el sistema																																
When busco vehículos con el valor "ABC" en la búsqueda																																
Then debería ver resultados de búsqueda con vehículos que coincidan con "ABC"																																
Scenario: Cambiar estado de un vehículo																																



```
Given existe un vehículo con id 1
When cambio el estado del vehículo con id 1 a "2"
Then debería ver el mensaje "Estado del vehículo cambiado correctamente."
```

- Análisis de los Feature
 - **Feature: Consulta de vehículos y detalles de vehículo en el sistema**
El sistema permite consultar la lista de vehículos activos.
El sistema muestra los detalles de un vehículo específico cuando se solicita.
 - **Feature: Registro de cliente**
El sistema permite registrar un nuevo cliente de manera exitosa.
El sistema no permite registrar un cliente que ya existe en la base de datos.
El sistema maneja errores al intentar registrar un nuevo cliente, por ejemplo, cuando falta algún dato obligatorio.
 - **Feature: Verificación de inicio de sesión en el sistema**
El sistema permite iniciar sesión con éxito utilizando un correo electrónico válido en la tabla de usuarios.
El sistema también permite iniciar sesión con éxito utilizando un correo electrónico válido en la tabla de usuarios (se repite el escenario).
El sistema no permite iniciar sesión con una contraseña incorrecta para el usuario administrador.
El sistema no permite iniciar sesión con una contraseña incorrecta para el usuario Ana.



- **En este punto podemos ver lo que logro pasar**

Se muestran un resumen de pruebas de software realizadas. Se muestra que se han ejecutado tres características diferentes: la consulta de vehículos, el registro de clientes y la verificación de inicio de sesión. Se ha realizado un total de 21 pasos de prueba, 7 escenarios y 3 características.



8. Pasos para hacer la Automatización de “Github Action”

GitHub Actions es una plataforma de integración continua y entrega continua (CI/CD) que permite automatizar flujos de trabajo de desarrollo de software directamente desde un repositorio de GitHub. Utiliza archivos YAML para definir procesos que se ejecutan en respuesta a eventos



GitHub Actions

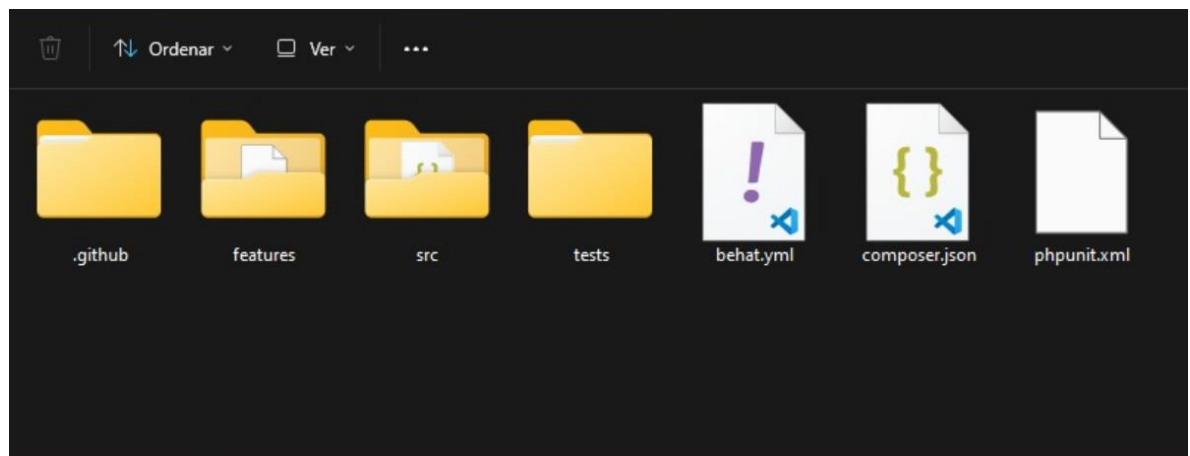
específicos como push, pull requests, issues, y más. Las características principales incluyen la capacidad de construir, probar y desplegar código, así como integrar con otros servicios y herramientas mediante el uso de acciones predefinidas o personalizadas. Esta herramienta facilita la automatización y mejora la eficiencia en el ciclo de vida del desarrollo del software.

- Se Realiza Clonación del proyecto

Se ha realizado un comando git clone para clonar un repositorio desde GitHub con la URL proporcionada. El proceso de clonación se ha completado con éxito, con todos los objetos necesarios descargados y procesados sin errores.

```
JOSE@JOSELUIS MINGW64 /c/REPOSITORIOS
$ git clone https://github.com/UPT-FAING-EPIS/proyecto-si784-2024-i-u3-pysistemaalquilerdevehiculos.git
Cloning into 'proyecto-si784-2024-i-u3-pysistemaalquilerdevehiculos'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

- Estructura de carpetas y archivos de un proyecto PHP: Esta es la estructura para proyectos PHP que utilizan tanto pruebas unitarias (PHPUnit) como pruebas de comportamiento (Behat) y que están configurados para la integración y entrega continua a través de GitHub Actions.



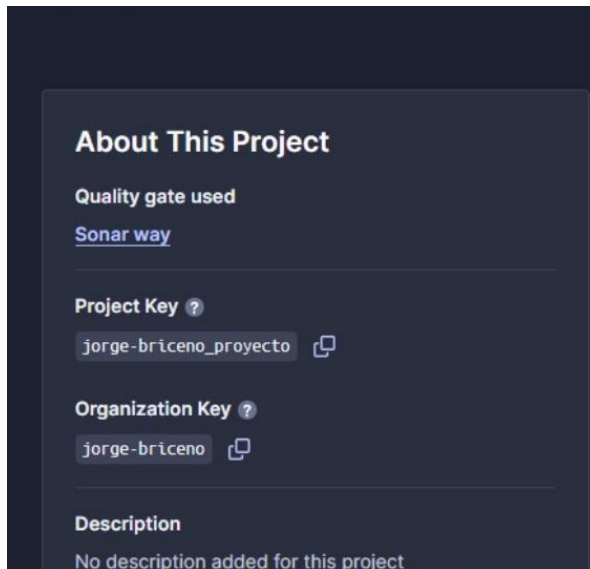
- Configuración del proyecto dentro de la plataforma de SonarQube
Ingresamos los datos requeridos

The screenshot shows the 'projects' configuration page in SonarQube. The 'Organization' dropdown is set to 'Jorge Briceño'. Below it, the 'Display Name' field is highlighted with a red box and contains the text 'PROYECTO'. The 'Project Key' field contains 'jorge-briceno_proyecto'. The 'Project visibility' section shows 'Public' selected. A 'Next' button is at the bottom.

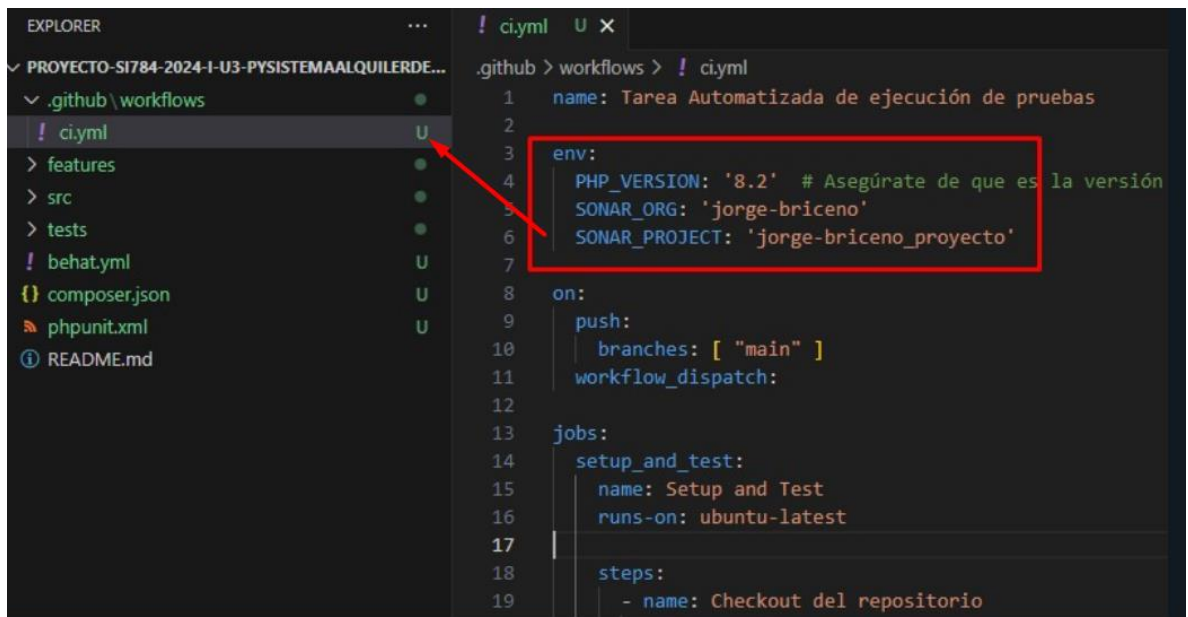
- La configuración de un proyecto en SonarQube bajo la metodología "Clean as You Code". Esta configuración permite definir qué parte del código se considerará como nuevo código: Seleccionamos "Previous version", lo que significa que se considera nuevo código cualquier cambio desde la versión anterior.

The screenshot shows the 'Set up project for Clean as You Code' page. It explains that the new code definition sets which part of the code will be considered new code. The 'Previous version' option is selected under 'The new code for this project will be based on:'. A red arrow points to the 'Previous version' radio button. The 'Number of days' option is also visible but not selected. A note at the bottom states: 'You can change this at any time in the project administration'.

- Esta sección proporciona una visión general rápida de la configuración y las claves importantes asociadas con el proyecto, así como la puerta de calidad utilizada para evaluar la calidad del código.



- El archivo YAML muestra cómo configurar un flujo de trabajo automatizado en GitHub Actions para ejecutar pruebas y realizar análisis de calidad de código con SonarQube. Esto asegura que el proyecto esté siempre bajo control de calidad continua y se mantenga en buen estado conforme se realizan cambios en el código.





- Se Muestra sus respectivos nombres, fechas de análisis y opciones de revocación. Uno de los elementos destacado es "ProyectoUnidad3"

Analyze "ProyectoTIENDA" 6	Never	25 April 2024	Revoke
Analyze "ProyectoUnidad3"	Never	11 July 2024	Revoke
Analyze "ProyectoUnidad3" 1	Never	11 July 2024	Revoke

- Sección de configuración de GitHub Pages para un repositorio. La sección resaltada indica que el sitio de GitHub Pages se está construyendo actualmente desde la rama "bddreporte".

Esta rama es la fuente del contenido para el sitio.

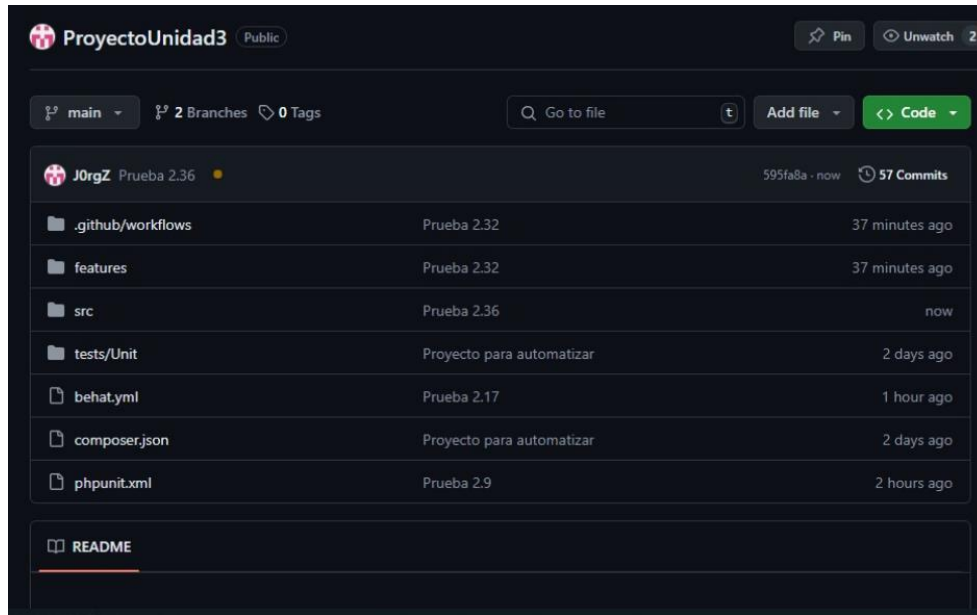
The screenshot shows the GitHub Pages configuration interface. On the left is a sidebar with navigation links: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages (selected), Security, Code security and analysis, Deploy keys, Secrets and variables, and Integrations. The main content area is titled 'GitHub Pages' and includes a description, a live site URL (<https://j0rgz.github.io/ProyectoUnidad3/>), and the 'Build and deployment' section. The 'Build and deployment' section is highlighted with a red box and shows the 'Source' dropdown set to 'Deploy from a branch' and the 'Branch' dropdown set to 'bddreporte'. Below the 'Branch' dropdown are buttons for 'Save' and 'Remove'. The 'Custom domain' section is also visible at the bottom.

- Se confirma que el sitio está en vivo y cualquier cambio reciente en el repositorio debería reflejarse en esa URL.

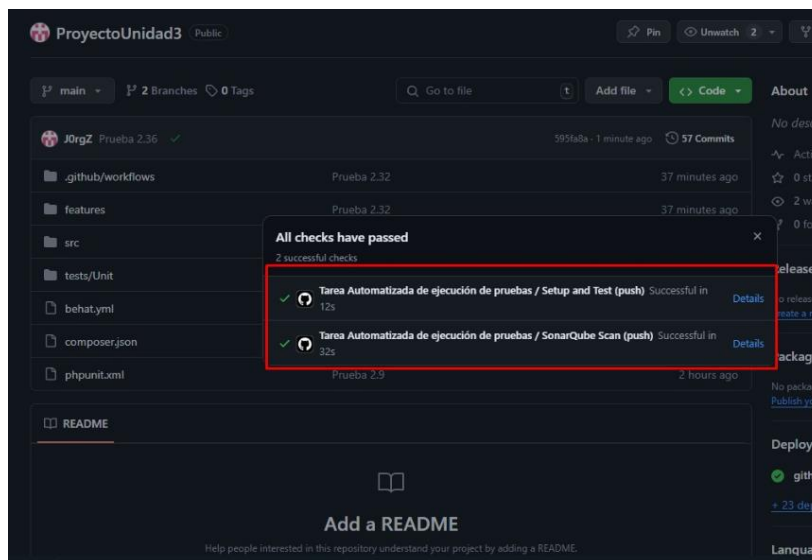
This is a close-up screenshot of the 'Build and deployment' section of the GitHub Pages configuration page. It shows the text 'Your site is live at <https://j0rgz.github.io/ProyectoUnidad3/>' and 'Last deployed by [j0rgz](#) 3 minutes ago'. The entire section is highlighted with a red box.



- Se encuentra en la rama principal (main), con 2 ramas en total y sin etiquetas. El repositorio tiene varias carpetas y archivos, como .github/workflows, features, src, tests/Unit, behat.yml, composer.json, y phpunit.xml.



- Se muestra una sección del repositorio de GitHub "ProyectoUnidad3" que indica que se han completado exitosamente dos verificaciones automáticas. Las tareas automáticas son "Setup and Test (push)" y "SonarQube Scan (push)", ambas etiquetadas como "Tarea Automatizada de ejecución de pruebas".





- Se muestra un registro de una acción de GitHub que se está ejecutando. La acción está destinada a desplegar un informe de Behat HTML a las páginas de GitHub. La acción comienza por obtener las entradas y configurar la autenticación, luego prepara los activos de publicación y clona el repositorio de GitHub.

```
Summary
Jobs
  Setup and Test
Run details
Usage
Workflow file

Setup and Test
Started 1h ago

Deploy Behat HTML report to GitHub Pages

20 Setup inputs
21 Setup auth token
22 Prepare publishing assets
23 [INFO] forcedOptions: false
24 /usr/bin/git clone --depth=1 --single-branch --branch=main https://github.com/ProjectUnidad3/ProjectUnidad3.git /home/runner/actions_github_pages_3720749783866
25 Cloning into '/home/runner/actions_github_pages_3720749783866'...
26 [INFO] clone -- /home/runner/actions_github_pages_3720749783866
27 [INFO] cd /home/runner/actions_github_pages_3720749783866
28 /usr/bin/git rm --ignore-unmatch *
29 rm *.nojekyll
30 rm 'index.html/assets/tw/css/callout.css'
31 rm 'index.html/assets/tw/css/callout.less'
32 rm 'index.html/assets/tw/css/style.css'
33 rm 'index.html/assets/tw/css/style.less'
34 rm 'index.html/index.html'
35 [INFO] cd /home/runner/actions_github_pages_3720749783866
36 [INFO] prepare publishing assets
37 [INFO] copy /home/runner/work/ProjectUnidad3/ProjectUnidad3/features/bootstrap/reports to /home/runner/actions_github_pages_3720749783866
38 cp: no such file or directory: /home/runner/work/ProjectUnidad3/ProjectUnidad3/features/bootstrap/reports/*
39 [INFO] delete excluded assets
40 rm -rf paths given
41 [INFO] Created /home/runner/actions_github_pages_3720749783866/.nojekyll
42 Setup Git config
43 Create a commit
44 Push the commit or tag
45 /usr/bin/git push origin main
46 To https://github.com/ProjectUnidad3/ProjectUnidad3.git
47 7b5d2b1..7b5d2b1 main -> main
48 [INFO] Action successfully completed

Post Checkout del repositorio
1 Post job cleanup.
2 /usr/bin/git version
3 git version 2.45.2
```

- Se muestra un registro de un análisis de SonarQube. Esta herramienta se utiliza para analizar la calidad del código y encontrar errores. Muestra los pasos involucrados en el proceso, como descargar archivos, configurar ajustes y cargar plugins.

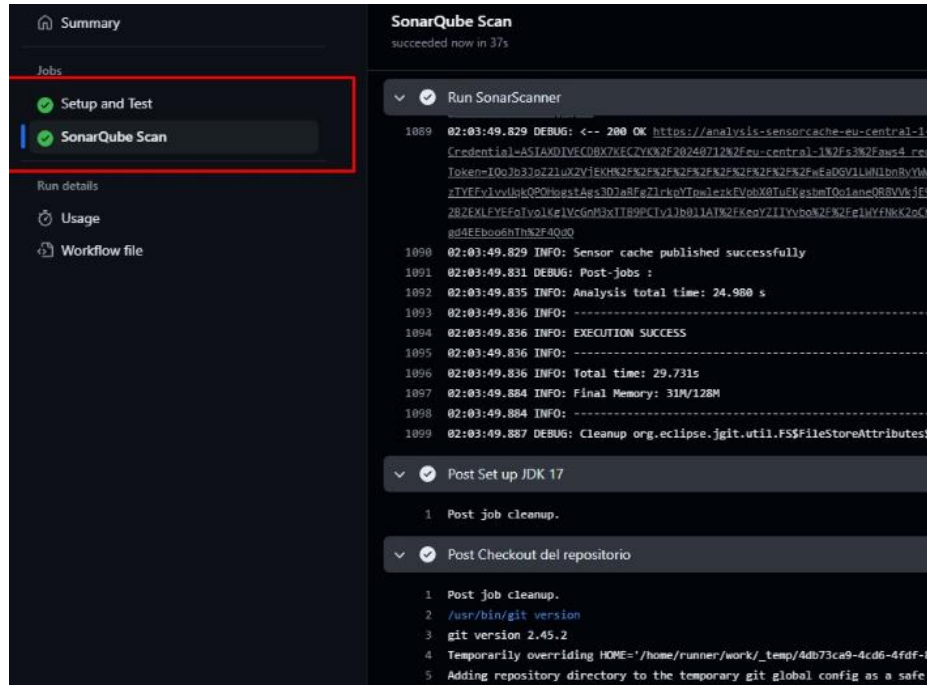
```
Summary
Jobs
  Setup and Test
  SonarQube Scan
Run details
Usage
Workflow file

SonarQube Scan
Started 9h ago

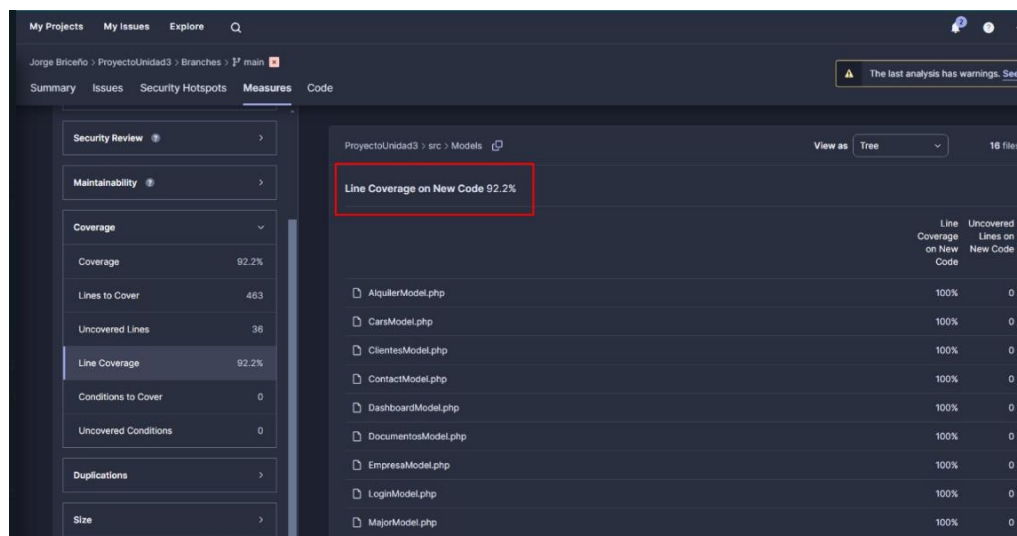
Run SonarScanner

33 02:03:20.665 DEBUG: Get bootstrap completed
34 02:03:20.667 DEBUG: Download https://sonarcloud.io/batch/file?name=scanner-sonarcloud-8.0.0.55884-all.jar to /home/runner/.sonar
35 02:03:21.541 DEBUG: Create isolated classloader...
36 02:03:21.549 DEBUG: Start temp cleaning...
37 02:03:21.551 DEBUG: Temp cleaning done
38 02:03:21.552 INFO: Analyzing on SonarCloud
39 02:03:21.552 INFO: Default locale: "en", source code encoding: "UTF-8" (analysis is platform dependent)
40 02:03:21.552 DEBUG: Work directory: /home/runner/work/ProjectUnidad3/ProjectUnidad3/.scannerwork
41 02:03:21.553 DEBUG: Execution execute
42 02:03:21.728 DEBUG: JVM max available memory: 3 GB
43 02:03:21.754 DEBUG: SonarCloud 10.9.0.2362
44 02:03:21.855 DEBUG: Sonar User Home: /home/runner/.sonar
45 02:03:21.855 DEBUG: Initialize DefaultScannerWsClient
46 02:03:22.059 DEBUG: Loaded [413] system trusted certificates
47 02:03:22.125 INFO: Load global settings
48 02:03:22.134 DEBUG: --> GET https://sonarcloud.io/api/settings/values?robotof
49 02:03:22.352 DEBUG: <-- 200 https://sonarcloud.io/api/settings/values?robotof (218s, unknown-length body)
50 02:03:22.359 INFO: Load global settings (done) | time=234ms
51 02:03:22.363 INFO: Server id: 180809FA-A648c9-T-TB3XqouHu
52 02:03:22.365 DEBUG: Initialize DownloadPluginsScannerWsClient
53 02:03:22.477 DEBUG: Loaded [413] system trusted certificates
54 02:03:22.481 DEBUG: Create : /home/runner/.sonar/.tmp
55 02:03:22.481 DEBUG: User cache: /home/runner/.sonar/cache
56 02:03:22.485 INFO: Loading required plugins
57 02:03:22.486 INFO: Load plugins index
58 02:03:22.486 DEBUG: --> GET https://sonarcloud.io/api/plugins/installed
59 02:03:22.892 DEBUG: <-- 200 https://sonarcloud.io/api/plugins/installed (315ms, unknown-length body)
60 02:03:22.895 INFO: Load plugins index (done) | time=349ms
61 02:03:22.895 INFO: Load/download plugins
62 02:03:22.836 DEBUG: Download plugin 'iac' to '/home/runner/.sonar/.tmp/fileCache13197022518250900972.tmp'
63 02:03:22.838 DEBUG: --> GET https://scanner-sonarcloud.io/plugins/iac/versions/4c1ba133faafe387cc4e45c8a76d2d8.jar
64 02:03:23.141 DEBUG: <-- 200 https://scanner-sonarcloud.io/plugins/iac/versions/4c1ba133faafe387cc4e45c8a76d2d8.jar (302ms, 1
65 02:03:23.541 DEBUG: Download plugin 'jacoco' to '/home/runner/.sonar/.tmp/fileCache7744953925468129750.tmp'
66 02:03:23.542 DEBUG: --> GET https://scanner-sonarcloud.io/plugins/jacoco/versions/04bba36a445c3713a372a358be2f99c.jar
```

- Se ve un resumen del proceso de integración continua para un proyecto de software. Se ve que se están ejecutando pruebas automáticas y se configura el entorno de desarrollo para el proyecto. Además, se muestra el estado de las pruebas y la configuración del repositorio de código.



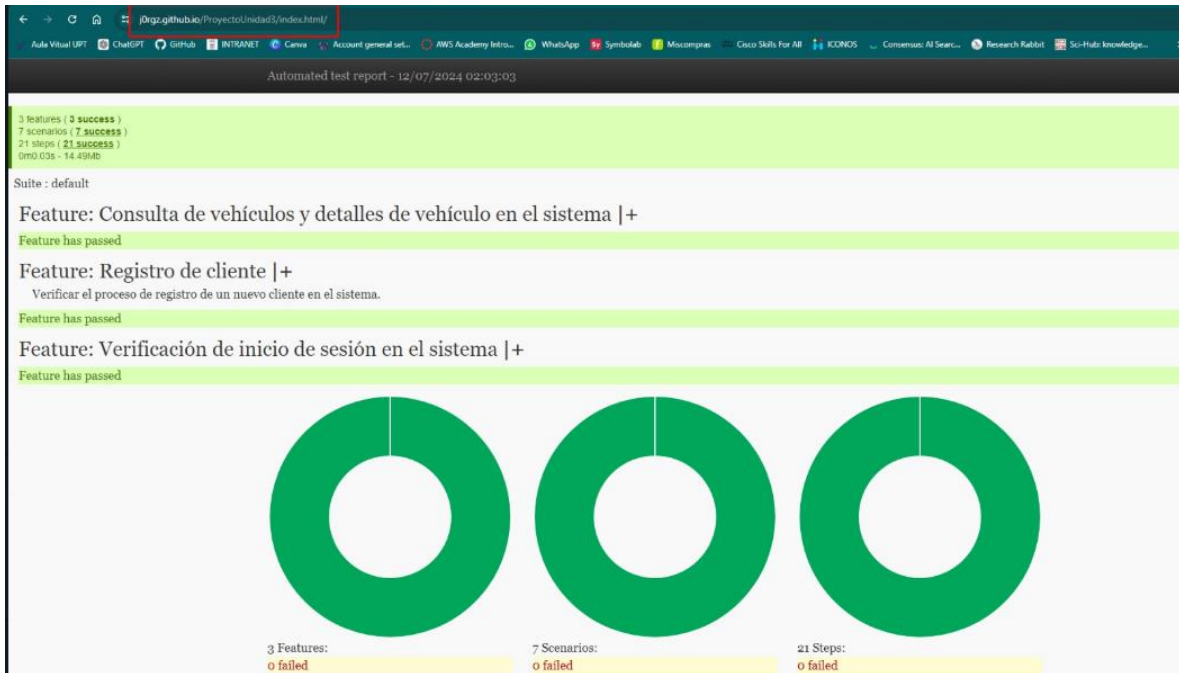
- Se usa una herramienta de análisis de código que muestra la cobertura de línea para un proyecto llamado "ProyectoUnidad3". La cobertura de línea es una medida de la cantidad de código que se ha probado. En este caso, la cobertura de línea para el código nuevo es del 92,2%.





Proyecto 2 Unidad - Aplicativo

- Se muestra un reporte automatizado de pruebas, alojado en GitHub Pages en la URL: <https://j0rgz.github.io/ProyectoUnidad3/index.html>. Este reporte demuestra que todas las pruebas automatizadas ejecutadas fueron exitosas, lo cual es un buen indicador de la calidad y estabilidad del sistema en su estado actual.



9. Cronograma

[illegible]

