

PROMETEO

Unidad 10: Pruebas, Depuración y Mantenimiento

Programación

Técnico Superior de DAM / DAW

```

12    }
13
14    }
15
16    }
17
18    }
19
20    }
21
22    }
23
24    }
25
26    }
27
28    }
29
30    }
31
32    }
33
34    }
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
779   }
780
781   }
782
783   }
784
785   }
786
787   }
788
789   }
789
790   }
791
792   }
793
794   }
795
796   }
797
798   }
799
800   }
801
802   }
803
804   }
805
806   }
807
808   }
809
809   }
810
811   }
812
813   }
814
815   }
816
817   }
818
819   }
819
820   }
821
822   }
823
824   }
825
826   }
827
828   }
829
829   }
830
831   }
832
833   }
834
835   }
836
837   }
838
839   }
839
840   }
841
842   }
843
844   }
845
846   }
847
848   }
849
849   }
850
851   }
852
853   }
854
855   }
856
857   }
858
859   }
859
860   }
861
862   }
863
864   }
865
866   }
867
868   }
869
869   }
870
871   }
872
873   }
874
875   }
876
877   }
878
879   }
879
880   }
881
882   }
883
884   }
885
886   }
887
888   }
889
889   }
890
891   }
892
893   }
894
895   }
896
897   }
898
899   }
899
900   }
900
901   }
901
902   }
902
903   }
903
904   }
904
905   }
905
906   }
906
907   }
907
908   }
908
909   }
909
910   }
910
911   }
911
912   }
912
913   }
913
914   }
914
915   }
915
916   }
916
917   }
917
918   }
918
919   }
919
920   }
920
921   }
921
922   }
922
923   }
923
924   }
924
925   }
925
926   }
926
927   }
927
928   }
928
929   }
929
930   }
930
931   }
931
932   }
932
933   }
933
934   }
934
935   }
935
936   }
936
937   }
937
938   }
938
939   }
939
940   }
940
941   }
941
942   }
942
943   }
943
944   }
944
945   }
945
946   }
946
947   }
947
948   }
948
949   }
949
950   }
950
951   }
951
952   }
952
953   }
953
954   }
954
955   }
955
956   }
956
957   }
957
958   }
958
959   }
959
960   }
960
961   }
961
962   }
962
963   }
963
964   }
964
965   }
965
966   }
966
967   }
967
968   }
968
969   }
969
970   }
970
971   }
971
972   }
972
973   }
973
974   }
974
975   }
975
976   }
976
977   }
977
978   }
978
979   }
979
980   }
980
981   }
981
982   }
982
983   }
983
984   }
984
985   }
985
986   }
986
987   }
987
988   }
988
989   }
989
990   }
990
991   }
991
992   }
992
993   }
993
994   }
994
995   }
995
996   }
996
997   }
997
998   }
998
999   }
999
1000   }
1000
1001   }
1001
1002   }
1002
1003   }
1003
1004   }
1004
1005   }
1005
1006   }
1006
1007   }
1007
1008   }
1008
1009   }
1009
1010   }
1010
1011   }
1011
1012   }
1012
1013   }
1013
1014   }
1014
1015   }
1015
1016   }
1016
1017   }
1017
1018   }
1018
1019   }
1019
1020   }
1020
1021   }
1021
1022   }
1022
1023   }
1023
1024   }
1024
1025   }
1025
1026   }
1026
1027   }
1027
1028   }
1028
1029   }
1029
1030   }
1030
1031   }
1031
1032   }
1032
1033   }
1033
1034   }
1034
1035   }
1035
1036   }
1036
1037   }
1037
1038   }
1038
1039   }
1039
1040   }
1040
1041   }
1041
1042   }
1042
1043   }
1043
1044   }
1044
1045   }
1045
1046   }
1046
1047   }
1047
1048   }
1048
1049   }
1049
1050   }
1050
1051   }
1051
1052   }
1052
1053   }
1053
1054   }
1054
1055   }
1055
1056   }
1056
1057   }
1057
1058   }
1058
1059   }
1059
1060   }
1060
1061   }
1061
1062   }
1062
1063   }
1063
1064   }
1064
1065   }
1065
1066   }
1066
1067   }
1067
1068   }
1068
1069   }
1069
1070   }
1070
1071   }
1071
1072   }
1072
1073   }
1073
1074   }
1074
1075   }
1075
1076   }
1076
1077   }
1077
1078   }
1078
1079   }
1079
1080   }
1080
1081   }
1081
1082   }
1082
1083   }
1083
1084   }
1084
1085   }
1085
1086   }
1086
1087   }
1087
1088   }
1088
1089   }
1089
1090   }
1090
1091   }
1091
1092   }
1092
1093   }
1093
1094   }
1094
1095   }
1095
1096   }
1096
1097   }
1097
1098   }
1098
1099   }
1099
1100   }
1100
1101   }
1101
1102   }
1102
1103   }
1103
1104   }
1104
1105   }
1105
1106   }
1106
1107   }
1107
1108   }
1108
1109   }
1109
1110   }
1110
1111   }
1111
1112   }
1112
1113   }
1113
1114   }
1114
1115   }
1115
1116   }
1116
1117   }
1117
1118   }
1118
1119   }
1119
1120   }
1120
1121   }
1121
1122   }
1122
1123   }
1123
1124   }
1124
1125   }
1125
1126   }
1126
1127   }
1127
1128   }
1128
1129   }
1129
1130   }
1130
1131   }
1131
1132   }
1132
1133   }
1133
1134   }
1134
1135   }
1135
1136   }
1136
1137   }
1137
1138   }
1138
1139   }
1139
1140   }
1140
1141   }
1141
1142   }
1142
1143   }
1143
1144   }
1144
1145   }
1145
1146   }
1146
1147   }
1147
1148   }
1148
1149   }
1149
1150   }
1150
1151   }
1151
1152   }
1152
1153   }
1153
1154   }
1154
1155   }
1155
1156   }
1156
1157   }
1157
1158   }
1158
1159   }
1159
1160   }
1160
1161   }
1161
1162   }
1162
1163   }
1163
1164   }
1164
1165   }
1165
1166   }
1166
1167   }
1167
1168   }
1168
1169   }
1169
1170   }
1170
1171   }
1171
1172   }
1172
1173   }
1173
1174   }
1174
1175   }
1175
1176   }
1176
1177   }
1177
1178   }
1178
1179   }
1179
1180   }
1180
1181   }
1181
1182   }
1182
1183   }
1183
1184   }
1184
1185   }
1185
1186   }
1186
1187   }
1187
1188   }
1188
1189   }
1189
1190   }
1190
1191   }
1191
1192   }
1192
1193   }
1193
1194   }
1194
1195   }
1195
1196   }
1196
1197   }
1197
1198   }
1198
1199   }
1199
1200   }
1200
1201   }
1201
1202   }
1202
1203   }
1203
1204   }
1204
1205   }
1205
1206   }
1206
1207   }
1207
1208   }
1208
1209   }
1209
1210   }
1210
1211   }
1211
1212   }
1212
1213   }
1213
1214   }
1214
1215   }
1215
1216   }
1216
1217   }
1217
1218   }
1218
1219   }
1219
1220   }
1220
1221   }
1221
1222   }
1222
1223   }
1223
1224   }
1224
1225   }
1225
1226   }
1226
1227   }
1227
1228   }
1228
1229   }
1229
1230   }
1230
1231   }
1231
1232   }
1232
1233   }
1233
1234   }
1234
1235   }
1235
1236   }
1236
1237   }
1237
1238   }
1238
1239   }
1239
1240   }
1240
1241   }
1241
1242   }
1242
1243   }
1243
1244   }
1244
1245   }
1245
1246   }
1246
1247   }
1247
1248   }
1248
1249   }
1249
1250   }
1250
1251   }
1251
1252   }
1252
1253   }
1253
1254   }
1254
1255   }
1255
1256   }
1256
1257   }
1257
1258   }
1258
1259   }
1259
1260   }
1260
1261   }
1261
1262   }
1262
1263   }
1263
1264   }
1264
1265   }
1265
1266   }
1266
1267   }
1267
1268   }
1268
1269   }
1269
1270   }
1270
1271   }
1271
1272   }
1272
1273   }
1273
1274   }
1274
1275   }
1275
1276   }
1276
1277   }
1277
1278   }
1278
1279   }
1279
1280   }
1280
1281   }
1281
1282   }
1282
1283   }
1283
1284   }
1284
1285   }
1285
1286   }
1286
1287   }
1287
1288   }
1288
1289   }
1289
1290   }
1290
1291   }
1291
1292   }
1292
1293   }
1293
1294   }
1294
1295   }
1295
1296   }
1296
1297   }
1297
1298   }
1298
1299   }
1299
1300   }
1300
1301   }
1301
1302   }
1302
1303   }
1303
1304   }
1304
1305   }
1305
1306   }
1306
1307   }
1307
1308   }
1308
1309   }
1309
1310   }
1310
1311   }
1311
1312   }
1312
1313   }
1313
1314   }
1314
1315   }
1315
1316   }
1316
1317   }
1317
1318   }
1318
1319   }
1319
1320   }
1320
1321   }
1321
1322   }
1322
1323   }
1323
1324   }
1324
1325   }
1325
1326   }
1326
1327   }
1327
1328   }
1328
1329   }
1329
1330   }
1330
1331   }
1331
1332   }
1332
1333   }
1333
1334   }
1334
1335   }
1335
1336   }
1336
1337   }
1337
1338   }
1338
1339   }
1339
1340   }
1340
1341   }
1341
1342   }
1342
1343   }
1343
1344   }
1344
1345   }
1345
1346   }
1346
1347   }
1347
1348   }
1348
1349   }
1349
1350   }
1350
1351   }
1351
1352   }
1352
1353   }
1353
1354   }
1354
1355   }
1355
1356   }
1356
1357   }
1357
1358   }
1358
1359   }
1359
1360   }
1360
1361   }
1361
1362   }
1362
1363   }
1363
1364   }
1364
1365   }
1365
1366   }
1366
1367   }
1367
1368   }
1368
1369   }
1369
1370   }
1370
1371   }
1371
1372   }
1372
1373   }
1373
1374   }
1374
1375   }
1375
1376   }
1376
1377   }
1377
1378   }
1378
1379   }
1379
1380   }
1380
1381   }
1381
1382   }
1382
1383   }
1383
1384   }
1384
1385   }
1385
1386   }
1386
1387   }
1387
1388   }
1388
1389   }
1389
1390   }
1390
1391   }
1391
1392   }
1392
1393   }
1393
1394   }
1394
1395   }
1395
1396   }
1396
1397   }
1397
1398   }
1398
1399   }
1399
1400   }
1400
1401   }
1401
1402   }
1402
1403   }
1403
1404   }
1404
1405   }
1405
1406   }
1406
1407   }
1407
1408   }
1408
1409   }
1409
1410   }
1410
1411   }
1411
1412   }
1412
1413   }
1413
1414   }
1414
1415   }
1415
1416   }
1416
1417   }
1417
1418   }
1418
1419   }
1419
1420   }
1420
1421   }
1421
1422   }
1422
1423   }
1423
1424   }
1424
1425   }
1425
1426   }
1426
1427   }
1427
1428   }
1428
1429   }
1429
1430   }
1430
1431   }
1431
1432   }
1432
1433   }
1433
1434   }
1434
1435   }
1435
1436   }
1436
1437   }
1437
1438   }
1438
1439   }
1439
1440   }
1440
1441   }
1441
1442   }
1442
1443   }
1443
1444   }
1444
1445   }
1445
1446   }
1446
1447   }
1447
1448   }
1448
1449   }
1449
1450   }
1450
1451   }
1451
1452   }
1452
1453   }
1453
1454   }
1454
1455   }
1455
1456   }
1456
1457   }
1457
1458   }
1458
1459   }
1459
1460   }
1460
1461   }
1461
1462   }
1462
1463   }
1463
1464   }
1464
1465   }
1465
1466   }
1466
1467   }
1467
1468   }
1468
1469   }
1469
1470   }
1470
1471   }
1471
1472   }
1472
1473   }
1473
1474   }
1474
1475   }
1475
1476   }
1476
1477   }
1477
1478   }
1478
1479   }
1479
1480   }
1480
1481   }
1481
1482   }
1482
1483   }
1483
1484   }
1484
1485   }
1485
1486   }
1486
1487   }
1487
1488   }
1488
1489   }
1489
1490   }
1490
1491   }
1491
1492   }
1492
1493   }
1493
1494   }
1494
1495   }
1495
1496   }
1496
1497   }
1497
1498   }
1498
1499   }
1499
1500   }
1500
1501   }
1501
1502   }
1502
1503   }
1503
1504   }
1504
1505   }
1505
1506   }
1506
1507   }
1507
1508   }
1508
1509   }
1509
1510   }
1510
1511   }
1511
1512   }
1512
1513   }
1513
1514   }
1514
1515   }
1515
1516   }
1516
1517   }
1517
1518   }
1518
1519   }
1519
1520   }
1520
1521   }
1521
1522   }
1522
1523   }
1523
1524   }
1524
1525   }
1525
1526   }
1526
1527   }
1527
1528   }
1528
1529   }
1529
1530   }
1530
1531   }
1531
1532   }
1532
1533   }
1533
1534   }
1534
1535   }
1535
1536   }
1536
1537   }
1537
1538   }
1538
1539   }
1539
1540   }
1540
1541   }
1541
1542   }
1542
1543   }
1543
1544   }
1544
1545   }
1545
1546   }
1546
1547   }
1547
1548   }
1548
1549   }
1549
1550   }
1550
1551   }
1551
1552   }
1552
1553   }
1553
1554   }
1554
1555   }
1555
1556   }
1556
1557   }
1557
1558   }
1558
1559   }
1559
1560   }
1560
1561   }
1561
1562   }
1562
1563   }
1563
1564   }
1564
1565   }
1565
1566   }
1566
1567   }
1567
1568   }
1568
1569   }
1569
1570
```

Esquema Visual



JUnit

- @Test
- Asserts
- @BeforeEach / @AfterEach



TestFX

- Interacciones GUI
- Clicks y escritura
- Pruebas end-to-end



Beneficios

- Calidad del software
- Evita regresiones
- Facilita refactorización

ⓘ Explicación del esquema

- **Pruebas en Java** es el concepto general que engloba tanto pruebas unitarias como pruebas funcionales.
- **JUnit** representa la parte lógica: estructura de test, ciclo de vida y validación mediante asserts.
- **TestFX** cubre la parte visual: simula acciones del usuario y valida que la interfaz responde adecuadamente.
- **Beneficios** resume el impacto directo: mayor calidad, reducción de errores y la posibilidad de evolucionar el software con seguridad.

Caso de Estudio

Un sistema de gestión interno basado en JavaFX

Una empresa tecnológica desarrollaba una herramienta interna utilizada por distintos departamentos para registrar pedidos, actualizar inventarios y generar reportes. La aplicación estaba construida con JavaFX, y aunque la lógica de negocio funcionaba correctamente, la interfaz gráfica sufría errores frecuentes: botones que no activaban los controladores, formularios que no validaban correctamente, escenas que no se refrescaban o tablas que no recogían los cambios tras una actualización.

Cada nueva versión introducía más problemas, y el equipo pasaba horas revisando la GUI manualmente. Esto ralentizaba la entrega de funcionalidades y generaba frustración en los usuarios internos.

Estrategia: Integración de JUnit + TestFX

01

Pruebas unitarias con JUnit para la lógica interna

Crear tests para validar:

- Cálculos en controladores.
- Reglas de negocio.
- Gestión de excepciones usando assertThrows.
- Procesos de validación de formularios.

02

Pruebas de interfaz con TestFX

Se diseñaron tests automatizados que simulaban:

- Un clic en el botón "Guardar".
- La escritura de datos en un formulario.
- La detección de errores al enviar formularios vacíos.
- El cambio correcto entre escenas sin lanzar excepciones.
- La actualización visual de tablas tras modificar datos.

03

Integración en CI/CD

Cada commit en Git ejecutaba automáticamente todas las pruebas. Si un test fallaba, la integración se bloqueaba hasta corregir el problema.

Resultado

En apenas dos semanas:

- Las incidencias de interfaz se redujeron drásticamente.
- Los desarrolladores ganaron confianza al refactorizar.
- El tiempo de validación manual disminuyó más del 70%.
- Los usuarios internos comenzaron a reportar menos errores.

El sistema de pruebas automatizadas se convirtió en un pilar del proceso de desarrollo y permitió que nuevas funcionalidades llegaran de forma estable y predecible.



Herramientas y Consejos

1

Mantén los tests claros, cortos y con un único propósito

Un test debe verificar un comportamiento concreto. Si hace demasiado, será difícil de entender y mantener.

2

Usa `assertThrows` para validar situaciones de error

Ejemplo típico:

```
assertThrows(IllegalArgumentException.class, () -> servicio.calcular(-1));
```

Esto demuestra que tu lógica maneja correctamente los casos límite.

3

En TestFX, evita depender de tiempos artificiales

Dormir el hilo con `Thread.sleep` genera tests inestables. Utiliza:

```
FxToolkit.setupStage(...)  
await().until(...)
```

para sincronizar acciones con la GUI.

4

Integra las pruebas en tu CI

Configura tu IDE o plataforma de CI (GitHub Actions, GitLab CI, Jenkins) para ejecutar todos los tests automáticamente al hacer commit o al crear un pull request. Esto evita que código defectuoso llegue a la rama principal.

5

Refactoriza con seguridad

Si un conjunto de tests está bien diseñado, cualquier cambio en la lógica romperá un test antes de llegar al usuario final.

Mitos y Realidades

 **Mito:** "Las pruebas ralentizan el desarrollo."

→ **FALSO.** Aunque al principio requieren tiempo, reducen errores, evitan regresiones y aceleran el desarrollo a medio plazo. El tiempo invertido en testear se recupera varias veces en mantenimiento.

 **Mito:** "TestFX es solo para probar botones."

→ **FALSO.** TestFX permite verificar navegación, validaciones visuales, actualización de escenas, efectos de animación y estados de la UI. Es una herramienta completa para pruebas funcionales end-to-end en JavaFX.

Resumen Final para Examen

- **JUnit** permite crear pruebas unitarias estructuradas con asserts y ciclo de vida controlado.
- **TestFX** automatiza interacciones reales con interfaces JavaFX (clics, escritura, navegación).
- Las pruebas combinadas reducen errores y evitan regresiones.
- Los asserts son el núcleo de la verificación en JUnit.
- Un buen conjunto de tests permite refactorizar con seguridad.



Sesión 25: Mantenimiento, control de versiones y documentación

Cuando un proyecto llega a producción, no termina su ciclo de vida: simplemente cambia de fase. A partir de ese momento, el software debe mantenerse vivo, estable y adaptable a nuevas necesidades. Esta etapa —el mantenimiento— es tan importante como la propia implementación inicial. Si no se gestiona correctamente, incluso la mejor aplicación puede volverse frágil, difícil de escalar y costosa de evolucionar.

Corrección de errores

El mantenimiento abarca varias actividades esenciales. La primera es la **corrección de errores**, que consiste en identificar fallos detectados por usuarios o por el equipo de QA y solucionarlos sin crear nuevos problemas. Esta tarea exige comprender el código existente, rastrear la causa original y validar que el comportamiento final es el esperado. A veces, un cambio aparentemente simple desencadena efectos secundarios; por eso, la comprensión profunda del sistema y la existencia de pruebas automatizadas son tan importantes.

Refactorización

Otra actividad clave es la **refactorización**, que no busca cambiar la funcionalidad, sino mejorar la estructura interna del código: eliminar duplicidades, renombrar métodos confusos, simplificar clases, dividir responsabilidades y mejorar el rendimiento. Refactorizar no es un capricho estético; es una estrategia imprescindible para evitar que el proyecto se degrade con el tiempo. Cuanto más claro y ordenado es el código, más fácil es mantenerlo, ampliarlo y reducir el riesgo de errores.

Documentación

En paralelo, un proyecto saludable requiere **documentación**. Esto no implica escribir manuales interminables, sino mantener actualizada la información mínima imprescindible: cómo ejecutar la aplicación, dependencias necesarias, estructura de carpetas, instrucciones para contribuir, decisiones técnicas relevantes y diagramas que clarifiquen la arquitectura. Una documentación clara evita que el proyecto dependa del conocimiento implícito de unos pocos desarrolladores. Cuando alguien se incorpora al equipo, la curva de aprendizaje disminuye drásticamente.

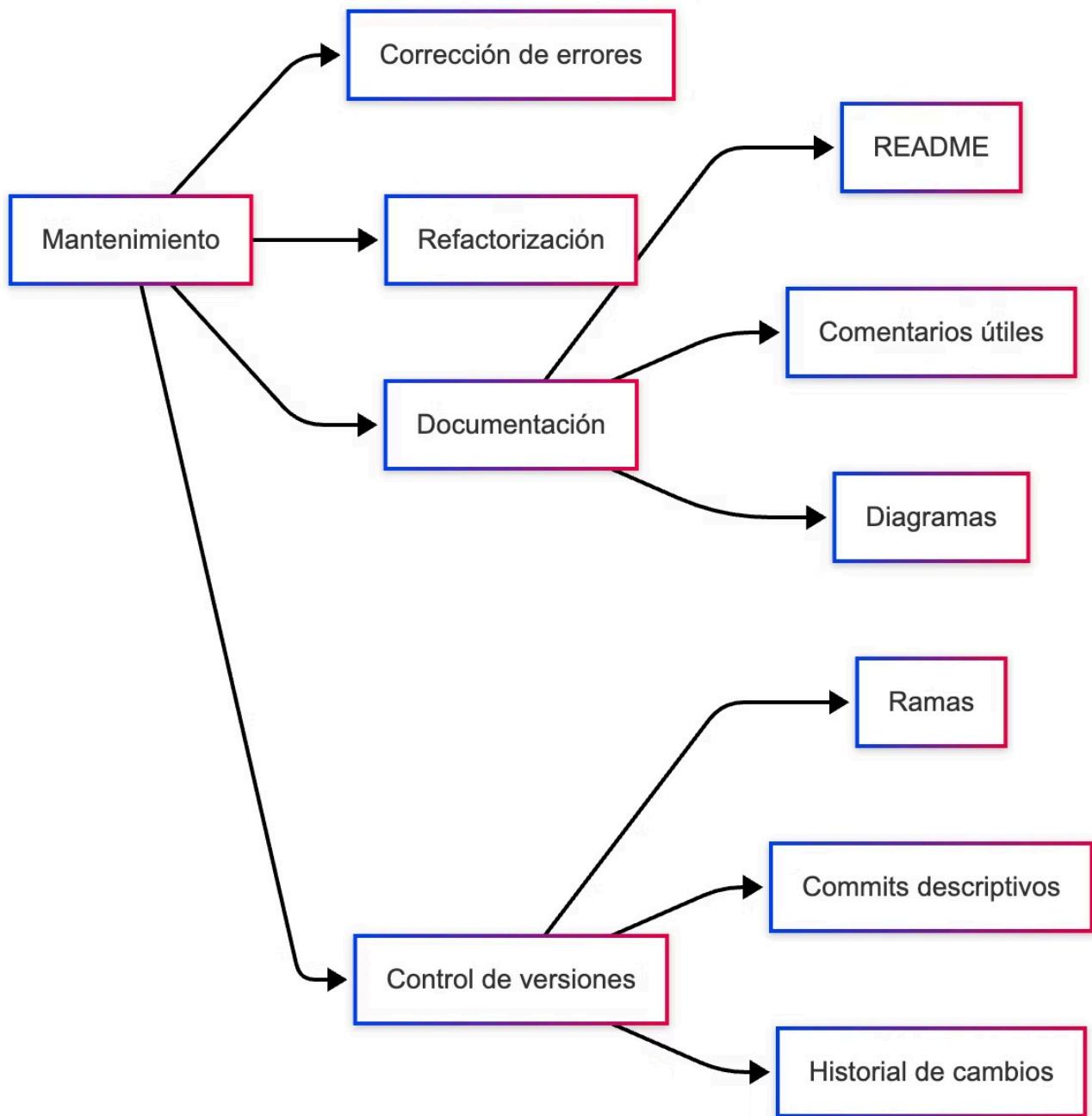
Control de versiones

Finalmente, el mantenimiento moderno sería prácticamente inviable sin un sistema de **control de versiones**, y en este terreno Git es el estándar absoluto. Git permite registrar cada cambio realizado en el código, recuperar versiones antiguas cuando algo falla, trabajar en ramas aisladas para evitar conflictos y coordinar a múltiples desarrolladores sin bloquearse mutuamente. Además, Git es la base de los flujos de integración continua, revisiones de código y despliegues automatizados, todos ellos imprescindibles para un software profesional.



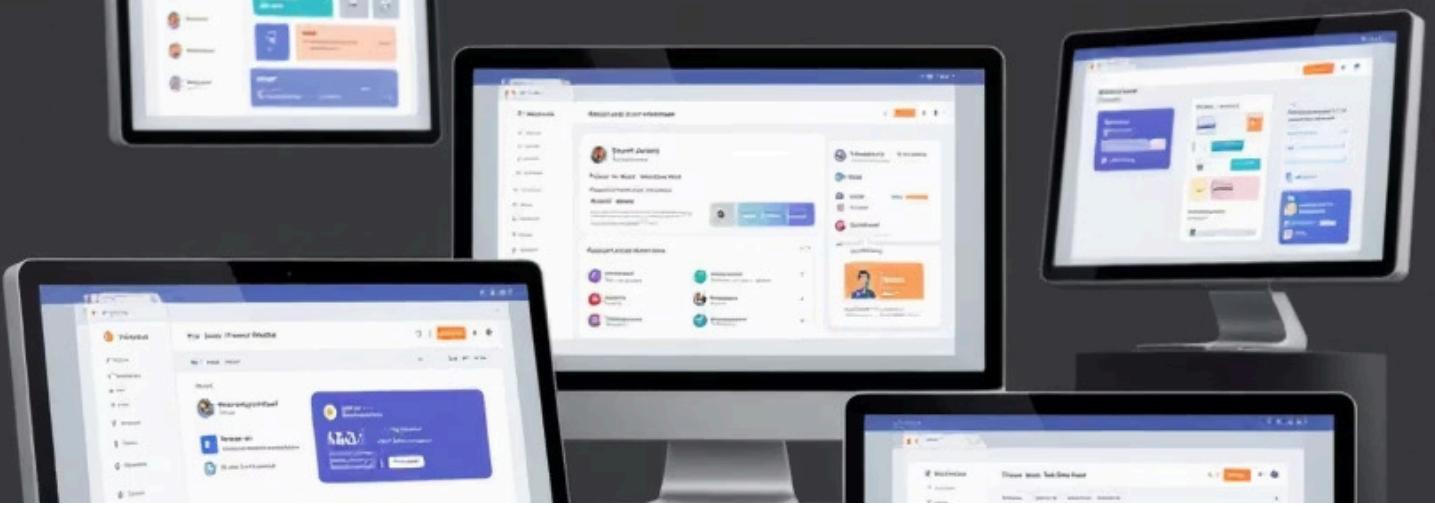
En conjunto, mantenimiento, documentación, refactorización y control de versiones forman un ecosistema que garantiza que el software siga siendo útil, estable y comprensible a lo largo de los años. Ignorar cualquiera de estas piezas aumenta la complejidad, reduce la calidad y eleva los costes del proyecto.

Esquema Visual



ⓘ Explicación del esquema

- **Mantenimiento** es el concepto central que engloba todas las actividades que mantienen vivo el proyecto.
- **Corrección de errores** gestiona incidencias detectadas en producción o pruebas.
- **Refactorización** mejora internamente el código sin alterar su comportamiento.
- **Documentación** incluye README, comentarios útiles y diagramas que permiten comprensión y continuidad.
- **Control de versiones** organiza cambios mediante ramas, commits y un historial que garantiza trazabilidad.



Caso de Estudio

Proyecto educativo con múltiples desarrolladores

Un equipo de desarrollo gestionaba una plataforma educativa dividida en varios módulos: autenticación, gestión de cursos, reportes de actividad y panel administrativo. La aplicación evolucionaba constantemente, pero el ritmo de crecimiento empezó a poner en evidencia problemas serios.

Contexto

Al revisar el repositorio, se detectaron varios patrones preocupantes:

Los commits se realizaban directamente sobre la rama principal sin revisión previa.

No existía un README más allá de un par de notas sueltas.

Los nombres de las ramas variaban entre desarrolladores, dificultando la organización.

El código duplicado crecía, especialmente en funciones de validación y carga de datos.

Los nuevos integrantes tardaban días en entender la estructura porque no existía documentación unificada. Los despliegues generaban conflictos y errores difíciles de rastrear. Algunos bugs se repetían porque no estaba claro qué cambios se habían aplicado en cada versión.

Estrategia implementada

01

Adopción de una estrategia de ramas clara

Se definió un flujo con ramas dedicadas:

- main para versiones estables
- develop para integración
- ramas de funcionalidad (feature/...)
- ramas de corrección (fix/...)

02

Commits descriptivos y obligatorios

Se acordó usar mensajes breves pero informativos:

- "fix: validar correo en registro"
- "refactor: extraer lógica de autenticación"

03

Revisiones obligatorias (pull requests)

Ningún cambio podía entrar en develop sin revisión de al menos una persona. Esto detectó errores antes de integrarlos.

04

Documentación mínima pero útil

Se creó un README con:

- instrucciones de instalación
- dependencias
- ejemplos de ejecución
- estructura de carpetas
- pasos para contribuir

También se añadieron diagramas simples de flujo y una guía de estilos de código.

Resultados

En menos de un mes:



- Las integraciones dejaron de generar conflictos críticos.
- Los errores se rastreaban con facilidad gracias al historial claro.
- Nuevos miembros podían comprender el proyecto en horas.
- La calidad del código mejoró gracias a la refactorización progresiva.

El equipo pasó de un desarrollo caótico a uno ordenado, predecible y sostenible.

Herramientas y Consejos

1

Realiza commits pequeños y descriptivos

Un commit debe representar una unidad de cambio coherente. Los mensajes deben explicar *qué* se cambió y *por qué*. Esto facilita revisiones y revertidos.

2

Mantén siempre un README actualizado

Incluye:

- instrucciones de instalación
- dependencias
- cómo ejecutar el proyecto
- comandos útiles
- estructura del repositorio

Un README actualizado evita cientos de dudas y acelera el onboarding.

3

Usa herramientas visuales para aprender Git

Clientes como GitKraken, Sourcetree o GitHub Desktop ayudan a entender el flujo visualmente, especialmente cuando empiezas.

4

Refactoriza en pasos pequeños y controlados

Antes de refactorizar:

- ejecuta los tests
- aplica cambios mínimos
- vuelve a ejecutar los tests

Esto evita introducir errores invisibles.

5

Documenta decisiones técnicas

Una breve explicación en un documento o comentario sobre por qué se eligió cierta arquitectura evita confusiones futuras.

Mitos y Realidades

X Mito: "Documentar es una pérdida de tiempo."

→ **FALSO.** Documentar reduce la dependencia de conocimiento implícito, acelera la incorporación de nuevos miembros y disminuye errores evitables. Un equipo sin documentación dedica más tiempo respondiendo dudas que desarrollando.

X Mito: "Refactorizar es peligroso porque puede romper el sistema."

→ **FALSO.** Refactorizar es seguro cuando existe una base sólida de pruebas automatizadas. De hecho, refactorizar reduce el riesgo futuro al mejorar la estructura del código.

❑ Resumen Final para Examen

- El mantenimiento incluye **corrección de errores, refactorización y documentación.**
- Git es esencial para gestionar cambios, colaborar y mantener un historial confiable.
- La documentación debe ser **breve, clara y actualizada.**
- Una estrategia de ramas bien definida reduce conflictos y facilita integración.
- Refactorizar mejora la calidad interna sin modificar la funcionalidad.

