

Herval Bernice Nganya Nana<sup>1</sup>

<sup>1</sup>*Fachbereich Informatik und Medien, Technische Hochschule Brandenburg, Deutschland  
nganyana@th-brandenburg.de*

Keywords: ...

Abriss: ...

Abstract: ...

## 1 PROTOTYP

In diesem Kapitel geht es um die Beschreibung des Weges zur Erzeugung eines Prototyps. Dieses wurde mittels der Werkzeuge Swagger (SmartBear, ) (Swagger UI, Swagger Editor und Swagger Codegen), Amazon Web Services und Hibernate erarbeitet. Für eine modellgetriebene Entwicklung wurde während der Durchführung dieses Projekts nicht Bottom-Up entwickelt, sondern Top-Down. Also wurde direkt von angefertigten Modellen zum generierten Produkt entwickelt.

### 1.1 Funktionalität

Da das Ziel des ganzen, ursprünglichen Projekts ein Anwendung zur Datenverwaltung ist, ermöglicht das in dieser Dokumentation beschriebene Rest-Service, der dafür verantwortlich ist Schnittstellen zu diesem Zweck bereitzustellen, folgende Funktionalitäten:

1. einen Benutzer zu erstellen
2. sich als Benutzer ein- und auszuloggen
3. Dateien hochzuladen, herunterzuladen, zu löschen, umzubenennen und aufzulisten
4. Ordner zu erstellen, zu löschen, umzubenennen und zu visualisieren.

Insgesamt sind von dem Rest-Service 11 Schnittstellen zur Verfügung gestellt.

### 1.2 Schnittstellenerzeugung

Der Prozess zur Erzeugung der Schnittstellen wurde in folgenden Schritten durchgeführt:

1. Zuerst wurden mittels des Werkzeugs Swagger Editor die Modelle, Schnittstellen sowie ein paar Meta-Daten des Projekts unter anderem der Titel, die Beschreibung, die Version beschrieben. All das wird entweder unter einer JSON- oder YAML-Datei gespeichert. In diesem Projekt wurde die JSON-Datei ausgewählt.
2. Anschließend wurde aus dieser Beschreibung eine graphische Dokumentation der Schnittstellen als Webseite mittels Swagger UI erzeugt. Die enthält einerseits die Schnittstellenbeschreibung, andererseits die Modellbeschreibung.
3. Dann wurde ebenfalls aus der JSON-Beschreibung der Quellcode anhand des Swagger-Codengen-Werkzeugs generiert. Swagger Codegen ermöglicht, den Quellcode in vielen Programmiersprachen (Backend sowie Frontend) zu erzeugen. Für dieses Projekt wurde ein Spring-Projekt gewählt, da den angestrebten Prototypen ein Java-basierter Rest-Service sein

soll.

4. Danach wurden in dem generierten Spring-Projekt Maven-Abhängigkeiten hinzugefügt. Diese sind unter anderem AWS S3, MySQL und Spring-Data.
5. Der nächste Schritt war dann die Verbindung zu der MySQL-Datenbank und die Modelle gemäß der verschiedenen Attribute, die in der Datenbank zu speichern sind, zu annotieren. Für diesen Schritt wurde Hibernate benutzt.
6. Abschließend wurde jede von Swagger automatisch generierte Funktion ausgefüllt. Bei der Generierung der Schnittstellen wird von Swagger eine leere Funktion pro Schnittstelle erzeugt. Es bleibt nur noch diese Funktionen auszufüllen.

### 1.3 Erweiterung der Schnittstellen

Der Rest-Service könnte noch Funktionalitäten bereitstellen, die in diesem Prototypen noch nicht entwickelt worden sind.

#### 1.3.1 Encryption-Möglichkeit

Das UserLogin-Modell kann um das Feld encryption erweitert werden. Dieses Feld ist so gedacht, dass die übertragenen Daten vor der Übertragung verschlüsselt werden. Bei True sollen die Daten chiffriert werden und bei False nicht. Über das Verschlüsselungsverfahren sollen sich die Entwickler entscheiden.

### 1.4 Rebuild des Projektes nach Schnittstellenbearbeitung

Das Projekt basiert auf einer Menge von Schritten, die nach und nach durchgeführt wurden. Gleich nach der Beschreibung der Modelle und der Schnittstellen bis zum generierten Projekt wird jede Code-Zeile automatisch generiert. Was ist denn, zu tun, falls Änderungen in der Beschreibung vorkommen? Der neu generierte Code soll integriert werden, ohne den bereits selbstgeschriebenen Code zu überschreiben. Zu diesem Zweck werden in dem Paper (Timo Greifenberg and Wortmann, 2015a) ein paar Methoden vorgestellt, die in solchen Fällen nützlich sind. Im Rahmen dieser Arbeit wurde aus Zeitgründen keine dieser Methoden implementiert, aber eine davon hat sich trotzdem durch die fünf von diesem Paper vorgestellten Kriterien (C1 bis C5) (Timo Greifenberg and Wortmann, 2015b) zu diesem Projekt passend gezeigt. Die heißt Generation Gap (Timo Greifenberg and Wortmann, 2015b).

Bei diesem Verfahren werden beispielsweise für eine Klasse NotePad ein Interface NotePad und eine Standard-Implementierung NotePadBaseImpl generiert. Diese Klasse zur Implementierung unterscheidet sich von der eigenen Implementierung, die zum Beispiel in der Klasse NotePadImpl geschrieben ist, in sofern als die dazugehörigen Codes unterschiedlich sind und die Klasse NotePadImpl wird zusätzlich bei einer neuen Erzeugung des Codes nicht überschrieben (Abbildung 1). NotePadImpl ist die Implementierung, die von beiden Codes (der generierte und der Selbstgeschriebene) verwendet werden soll.

Gründe für die Wahl dieser Methode sind:

1. Die Struktur des endgültigen Quellcodes in diesem Projekt weicht nur von dem ursprünglichen Code um die Pakete io.swagger.service, io.swagger.repository, io.swagger.configuration. Ähnlich NotePadImpl in (Timo Greifenberg and Wortmann, 2015b) sind die Klassen wie FileService, FolderService und UserService die selbstgeschriebenen Klassen, die nicht überschrieben werden sollten. Dies erfüllt das Kriterium C1.
2. Die anderen Klassen in io.swagger.api, io.swagger.model und io.swagger sind hier generierte Pakete. Ihre Inhalte können beliebig überschrieben werden. Das Kriterium C2 ist erfüllt.
3. Die generierten Schnittstellen können im Laufe des Projekts erweitert werden, ohne den selbstgeschriebenen Code zu beeinflussen. Das Kriterium C3 ist erfüllt.
4. Das Kriterium C4 ist ebenfalls erfüllt, da der Selbstgeschriebene Code unabhängig von dem generierten Code ist. Dies ist auf die Tatsache, dass der generierte Code keinen Einfluss auf den Selbstgeschriebenen hat.
5. In diesem Projekt wurde der Code in Spring (Java) generiert. Nach einer neuen Erzeugung ist der generierte Code immernoch auf die gleiche Programmiersprache. Dies fordert deshalb keine zusätzliche Programmiersprache. So ist das Kriterium C5 erfüllt.

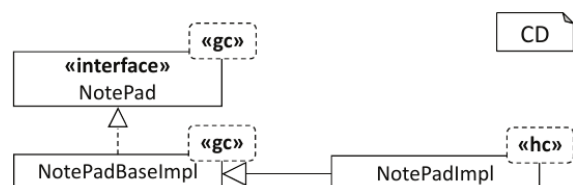


Figure 1: Generation Gap Muster für das NotePad-Beispiel  
gc: generated code, hc: hand code

## 2 Ergebnisse und Probleme

### 2.1 Ergebnisse

Schließlich liegt nach der Arbeitsweise der gezielte Rest-Service vor. Dieser ist in der Lage die gewünschten Funktionalitäten anzubieten, die vorher abgesprochen waren. Die Daten sind durch die Datenbank und Speicherung der Daten auf einer AWS-S3-Instanz tatsächlich persistent. Der Rest-Service kann ebenfalls entsprechende Antworten zurückgeben:

1. 200 OK
2. 400 Bad Request
3. 403 Forbidden
4. 404 Not Found

### 2.2 Probleme

Während der Entwicklung ist das Team auf Schwierigkeiten gestoßen. Diese waren einerseits aufgrund der Verwendung eines Modellgetriebene-Software-Entwicklungswerkzeugs ausgelöst andererseits aufgrund der Datensicherheit bei der Nutzung vom AWS S3.

#### 2.2.1 Unübersichtlicher, generierter Code

Der von Swagger generierte Code ist an vielen Stellen unübersichtlich. Der Code in den Controllern und Interfaces ist schwer zu verstehen (Abbildung 2).

```
115 @ApiOperation(value = "Edit a folder", notes = "This can only be done by the logged in user.", response = Folder.class, tags = {"folders"})
116 @ApiResponses(value = {
117     @ApiResponse(code = 200, message = "Successful operation", response = Folder.class),
118     @ApiResponse(code = 401, message = "Unauthorized", response = Folder.class),
119     @ApiResponse(code = 404, message = "Not found", response = Folder.class) })
120 @RequestMapping(value = "/{token}/{folderId}",
121     produces = { "application/json" },
122     method = RequestMethod.PUT)
123 ResponseEntity<Folder> editFolder(
124     @ApiParam(value = "Id of the folder to edit", required=true ) @PathVariable("folderId") String folderId
125 )
126
127
128
129
130
131
132
133 @ApiParam(value = "Token of the current user", required=true ) @PathVariable("token") String token
134
135
136
137
138
139
140
141 @ApiParam(value = "Edit a folder in data storage", required=true ) @RequestBody FolderCreate folder
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158 @ApiOperation(value = "Get a file", notes = "This can only be done by the logged in user.", response = File.class, tags = {"files"})
159 @ApiResponses(value = {
160     @ApiResponse(code = 200, message = "Successful operation", response = File.class),
161     @ApiResponse(code = 401, message = "Unauthorized", response = File.class),
162     @ApiResponse(code = 404, message = "Not found", response = File.class) })
163 @RequestMapping(value = "/{token}/{folderId}/files",
164     produces = { "application/json" },
165     method = RequestMethod.GET)
166 ResponseEntity<File> getFile(
167     @ApiParam(value = "Token of the current user", required=true ) @PathVariable("token") String token
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
```

Figure 2: Quellcode der Schnittstellen getFile und editFile

#### 2.2.2 Projektstruktur

Jeder Entwickler und jede Organisation können ihre eigene Paketstruktur besitzen. Daher ist die von

Swagger angebotene Paketstruktur immer auf dem ersten Blick ungewöhnlich (Abbildung 3). In dieser Abbildung ist das Paket io.swagger.service nicht von Swagger sondern von dem Entwicklungsteam generiert.



Figure 3: Von Swagger generierte und vom Entwicklungsteam veränderte Paketstruktur

#### 2.2.3 Sicherheit

AWS bietet den S3-Dienst an, der ermöglicht Dateien auf eine Cloud zu speichern. Dies erfolgt per Quellcode mittels Access- und Secret-Keys. Damit das Team Transaktionen mit S3 während der Entwicklung durchführen kann, soll es diese Schlüssel besitzen. Das Problem an der Stelle ist die Verteilung dieser Schlüssel. Diese kann zu bösen Zwecken verwendet werden und schließlich hohe Kosten für den offiziellen Besitzer verursachen. Außerdem wären

sie öffentlich für Hacker durch GitHub gewesen, da GitHub zur Versionsverwaltung verwendet wurde. Aus diesen Gründen wurde ein Nexus-Repository entwickelt und geheim gehalten, damit diese Access- und Secret-Keys nicht verteilt werden. Dieses Repository stellt ein Object S3Transaction zur Verfügung, das seinerseits Funktionen wie upload, rename und delete zur Verfügung stellt. Die S3Transaction-Abhängigkeit ist anhand der folgenden Dependency aufzurufen:

```
<dependency>
<groupId>
com.mdsd-2017-2018.s3-transactions
</groupId>
<artifactId>
s3-transactions
</artifactId>
<version>1.2.2</version>
</dependency>
```

#### 2.2.4 Abstürzen der EC2-Instanz

## REFERENCES

- SmartBear. Swagger - world's most popular api framework.
- Timo Greifenberg, Katrin Hlldobler, C. K. M. L. P. M. S. N. K. M. A. N. P. D. P. D. R. A. R. B. R. M. S. and Wortmann, A. (2015a). *A Comparison of Mechanisms for Integrating Handwritten and Generated Code for Object-Oriented Programming Languages*. Software Engineering, RWTH Aachen University, Institute for Building Services and Energy Design, TU Braunschweig.
- Timo Greifenberg, Katrin Hlldobler, C. K. M. L. P. M. S. N. K. M. A. N. P. D. P. D. R. A. R. B. R. M. S. and Wortmann, A. (2015b). *A Comparison of Mechanisms for Integrating Handwritten and Generated Code for Object-Oriented Programming Languages*, Page 76, Chapter 3.1. Software Engineering, RWTH Aachen University, Institute for Building Services and Energy Design, TU Braunschweig.

## List of Tables

## List of Figures