

2024-2025 YILI BAHAR DÖNEMİ
BULUT BİLİŞİM VE UYGULAMALARI
PROJE ÖDEVLERİ TESLİM DOKÜMANI



AD – SOYAD

BORA KOCABIYIK:

HÜSEYİN TINAZTEPE:

YUSUF TUNÇ:

ÖĞRENCİ NUMARASI

21290270

21290360

22290071

GitHub Linki: <https://github.com/brckfrc/cloud-projeleri/>

PROJE 1 - YouTube Linki: Çift Katmanlı Web Uygulaması (Web API + Frontend)

Bora - <https://youtu.be/EyVPyuHyZCw>

Hüseyin - <https://youtu.be/uzP7cAMsIx4>

Yusuf - https://youtu.be/2Uld3BLzK_E

PROJE 2 - YouTube Linki: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

Bora - <https://youtu.be/P8G7fpsn-ig>

Hüseyin - <https://youtu.be/Lqkzts8Yn48>

Yusuf - <https://youtu.be/hDo2oQ2puFU>

PROJE 3 - YouTube Linki: E-Ticaret Uygulaması (Otomatik Ölçeklendirme ve Yönetim)

Bora - <https://youtu.be/K70y9Q2krTs>

Hüseyin - <https://youtu.be/9z-mOofC4VY>

Yusuf - <https://youtu.be/Lglhx8VGsTo>

PROJE 4 - YouTube Linki: Gerçek Zamanlı Veri Akışı ve İşleme (IoT Uygulaması)

Bora - <https://youtu.be/tA8p7Vt9Aio>

Hüseyin - <https://youtu.be/nQBN8iOe3TE>

Yusuf - <https://youtu.be/ZfUPqlrDnx8>

⚠️ Not: MongoDB veritabanını dış erişime açmak için gerekli kullanıcı yetkilerini vermeye çalıştım; ancak güvenlik nedeniyle kendi hesabımı paylaşmam mümkün olmadı. Ayrıca servis hesabına ait JSON dosyasını da dışa aktaramadım. Bu nedenle doğrudan bağlantı veremedim. Ancak sistemin tüm işleyişi ve veritabanı entegrasyonu video sunumunda ayrıntılı şekilde gösterilmiştir.

Ayrıca, Proje 1 ve Proje 3, kullanıcı arayüzü odaklı web projeleri olduğu için sabit bir veri seti kullanılmamıştır. Proje 2'de kullanılan veri seti klasöre eklenmiştir. Proje 4'te kullanılan veriler ise Python scripti ile anlık olarak üretilmekte olup, sabit bir dosya biçiminde bulunmamaktadır.

PROJE 1: Çift Katmanlı Web Uygulaması (Web API + Frontend)

SkyNotes Tam Yıgın Not Yönetim Uygulaması Proje Raporu

1. Proje Genel Bakış

1.1 Proje Amacı

SkyNotes, kullanıcıların dijital not yönetimini kolaylaştırmak için geliştirilmiş bir web uygulamasıdır. Proje, modern web geliştirme teknolojilerini kullanarak kullanıcı dostu bir not yönetim sistemi oluşturmayı hedeflemektedir.

2. Teknoloji Yığını

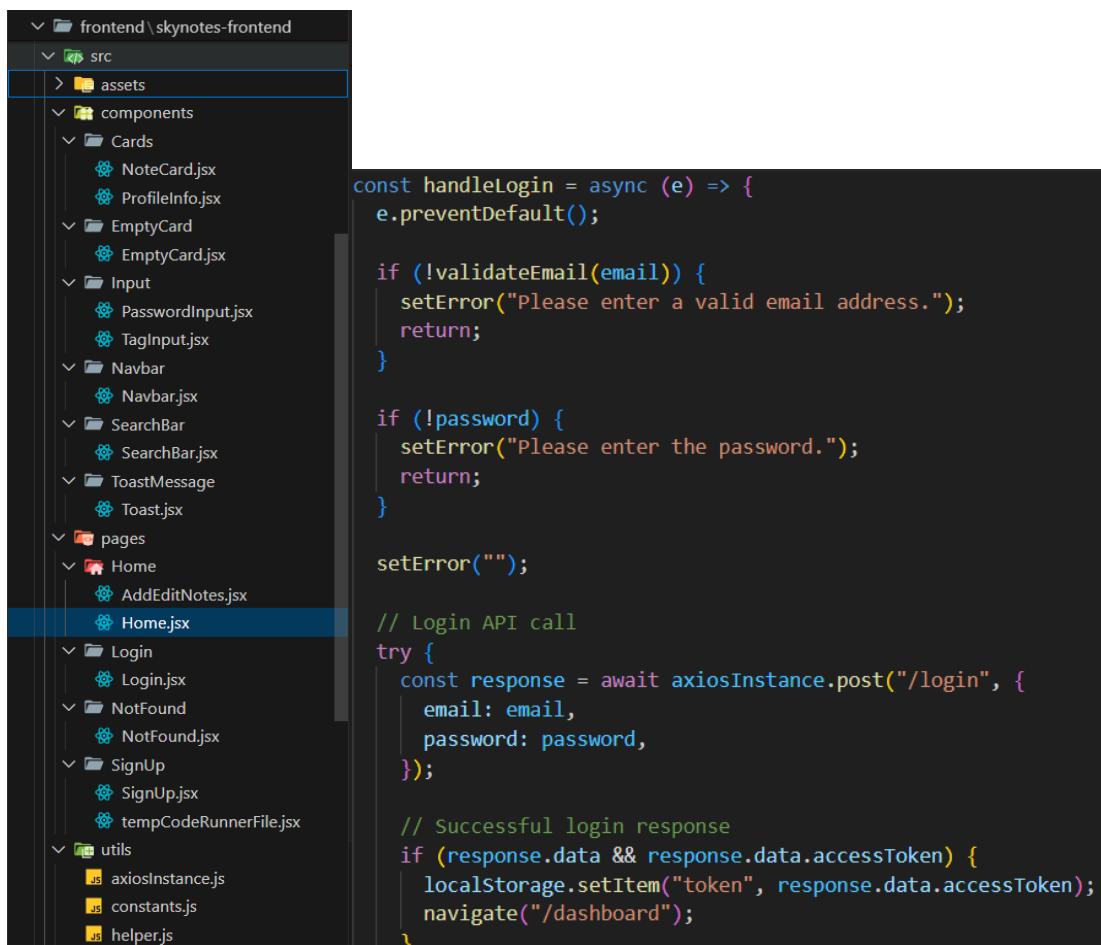
2.1 Frontend Teknolojileri

SkyNotes'un frontend geliştirme süreci, kullanıcı deneyimini merkeze alan bir yaklaşım ile ilerletildi. Proje başlangıcında Vite ile hızlı ve modern bir React projesi kurularak temeller atıldı. İlk olarak, uygulamanın genel mimari yapısı ve bileşen hiyerarşisi planlandı. React Router DOM kullanılarak sayfa navigasyonu için sağlam bir altyapı oluşturuldu.

Tailwind CSS ile stil oluşturma süreci, hızlı ve responsive bir tasarım sağlamak üzere yapılandırıldı. Her bileşen, kullanıcı deneyimini optimize edecek şekilde detaylı olarak tasarlandı. Geliştirme sürecinde sürekli olarak bileşenlerin yeniden kullanılabilirliği ve kod tekrarından kaçınma prensipleri gözetildi.

- React: Kullanıcı arayüzü geliştirmek için modern JavaScript kütüphanesi
- React Router DOM: Sayfa yönlendirme ve navigasyon için
- Tailwind CSS: Hızlı ve esnek stil oluşturma için utility-first CSS framework'ü
- React Icons: Kullanıcı arayüzü için zengin icon seti

```
14 const Home = () => {
15   const [openAddEditModal, setOpenAddEditModal] = useState({
16     isShown: false,
17     type: "add",
18     data: null,
19   });
20
21   const [showToastMsg, setShowToastMsg] = useState({
22     isShown: false,
23     message: "",
24     type: "add",
25   });
26
27   const [allNotes, setAllNotes] = useState([]);
28   const [userInfo, setUserInfo] = useState(null);
29   const [loading, setLoading] = useState(true);
30   const [isSearch, setSearch] = useState(false);
31
32   const navigate = useNavigate();
33
34 > const handleEdit = (noteDetails) => {
35   };
36
37 > const showToastMessage = (message, type) => {
38   };
39
40 > const handleCloseToast = () => {
41   };
42
43 > const handleClearSearch = () => {
44   };
45
46 > const handlePinnedNote = (noteData) => {
47   };
48
49 > const handleSearchNote = (query) => {
50   };
51
52 > const handleGetAllNotes = () => {
53   // Get user info
54 >   const getUserInfo = async () => {
55     ...
56   };
57
58 >   // Get all notes
59 >   const getAllNotes = async () => {
60     ...
61   };
62
63 >   // Delete note
64 >   const deleteNote = async (data) => {
65     ...
66   };
67
68 >   // Search note
69 >   const onSearchNote = async (query) => {
70     ...
71   };
72
73 >   // Pinned note
74 >   const updateIsPinned = async (noteData) => {
75     ...
76   };
77
78 >   // Clear search
79 >   const handleClearSearch = () => {
80     ...
81   };
82
83 >   // Effect
84 >   useEffect(() => {
85     ...
86   }, []);
87
88 >   // Get user info
89 >   const getUserInfo = async () => {
90     ...
91   };
92
93 >   // Get all notes
94 >   const getAllNotes = async () => {
95     ...
96   };
97
98 >   // Delete note
99 >   const deleteNote = async (data) => {
100    ...
101  };
102
103 >   // Search note
104 >   const onSearchNote = async (query) => {
105    ...
106  };
107
108 >   // Pinned note
109 >   const updateIsPinned = async (noteData) => {
110    ...
111  };
112
113 >   // Clear search
114 >   const handleClearSearch = () => {
115     ...
116   };
117
118 >   // Effect
119 >   useEffect(() => {
120     ...
121   }, []);
122
123 >   // Get user info
124 >   const getUserInfo = async () => {
125     ...
126   };
127
128 >   // Get all notes
129 >   const getAllNotes = async () => {
130     ...
131   };
132
133 >   // Delete note
134 >   const deleteNote = async (data) => {
135     ...
136   };
137
138 >   // Search note
139 >   const onSearchNote = async (query) => {
140     ...
141   };
142
143 >   // Pinned note
144 >   const updateIsPinned = async (noteData) => {
145     ...
146   };
147
148 >   // Clear search
149 >   const handleClearSearch = () => {
150     ...
151   };
152
153 >   // Effect
154 >   useEffect(() => {
155     ...
156   }, []);
157
158 >   // Get user info
159 >   const getUserInfo = async () => {
160     ...
161   };
162
163 >   // Get all notes
164 >   const getAllNotes = async () => {
165     ...
166   };
167
168 >   // Delete note
169 >   const deleteNote = async (data) => {
170     ...
171   };
172
173 >   // Search note
174 >   const onSearchNote = async (query) => {
175     ...
176   };
177
178 >   // Pinned note
179 >   const updateIsPinned = async (noteData) => {
180     ...
181   };
182
183 >   // Clear search
184 >   const handleClearSearch = () => {
185     ...
186   };
187
188 >   // Effect
189 >   useEffect(() => {
190     ...
191   }, []);
192
193 >   // Get user info
194 >   const getUserInfo = async () => {
195     ...
196   };
197
198 >   // Get all notes
199 >   const getAllNotes = async () => {
200     ...
201   };
202
203 >   // Delete note
204 >   const deleteNote = async (data) => {
205     ...
206   };
207
208 >   // Search note
209 >   const onSearchNote = async (query) => {
210     ...
211   };
212
213 >   // Pinned note
214 >   const updateIsPinned = async (noteData) => {
215     ...
216   };
217
218 >   // Clear search
219 >   const handleClearSearch = () => {
220     ...
221   };
222
223 >   // Effect
224 >   useEffect(() => {
225     ...
226   }, []);
227
228 >   // Get user info
229 >   const getUserInfo = async () => {
230     ...
231   };
232
233 >   // Get all notes
234 >   const getAllNotes = async () => {
235     ...
236   };
237
238 >   // Delete note
239 >   const deleteNote = async (data) => {
240     ...
241   };
242
243 >   // Search note
244 >   const onSearchNote = async (query) => {
245     ...
246   };
247
248 >   // Pinned note
249 >   const updateIsPinned = async (noteData) => {
250     ...
251   };
252
253 >   // Clear search
254 >   const handleClearSearch = () => {
255     ...
256   };
257
258 >   // Effect
259 >   useEffect(() => {
260     ...
261   }, []);
262
263 >   // Get user info
264 >   const getUserInfo = async () => {
265     ...
266   };
267
268 >   // Get all notes
269 >   const getAllNotes = async () => {
270     ...
271   };
272
273 >   // Delete note
274 >   const deleteNote = async (data) => {
275     ...
276   };
277
278 >   // Search note
279 >   const onSearchNote = async (query) => {
280     ...
281   };
282
283 >   // Pinned note
284 >   const updateIsPinned = async (noteData) => {
285     ...
286   };
287
288 >   // Clear search
289 >   const handleClearSearch = () => {
290     ...
291   };
292
293 >   // Effect
294 >   useEffect(() => {
295     ...
296   }, []);
297
298 >   // Get user info
299 >   const getUserInfo = async () => {
300     ...
301   };
302
303 >   // Get all notes
304 >   const getAllNotes = async () => {
305     ...
306   };
307
308 >   // Delete note
309 >   const deleteNote = async (data) => {
310     ...
311   };
312
313 >   // Search note
314 >   const onSearchNote = async (query) => {
315     ...
316   };
317
318 >   // Pinned note
319 >   const updateIsPinned = async (noteData) => {
320     ...
321   };
322
323 >   // Clear search
324 >   const handleClearSearch = () => {
325     ...
326   };
327
328 >   // Effect
329 >   useEffect(() => {
330     ...
331   }, []);
332
333 >   // Get user info
334 >   const getUserInfo = async () => {
335     ...
336   };
337
338 >   // Get all notes
339 >   const getAllNotes = async () => {
340     ...
341   };
342
343 >   // Delete note
344 >   const deleteNote = async (data) => {
345     ...
346   };
347
348 >   // Search note
349 >   const onSearchNote = async (query) => {
350     ...
351   };
352
353 >   // Pinned note
354 >   const updateIsPinned = async (noteData) => {
355     ...
356   };
357
358 >   // Clear search
359 >   const handleClearSearch = () => {
360     ...
361   };
362
363 >   // Effect
364 >   useEffect(() => {
365     ...
366   }, []);
367
368 >   // Get user info
369 >   const getUserInfo = async () => {
370     ...
371   };
372
373 >   // Get all notes
374 >   const getAllNotes = async () => {
375     ...
376   };
377
378 >   // Delete note
379 >   const deleteNote = async (data) => {
380     ...
381   };
382
383 >   // Search note
384 >   const onSearchNote = async (query) => {
385     ...
386   };
387
388 >   // Pinned note
389 >   const updateIsPinned = async (noteData) => {
390     ...
391   };
392
393 >   // Clear search
394 >   const handleClearSearch = () => {
395     ...
396   };
397
398 >   // Effect
399 >   useEffect(() => {
400     ...
401   }, []);
402
403 >   // Get user info
404 >   const getUserInfo = async () => {
405     ...
406   };
407
408 >   // Get all notes
409 >   const getAllNotes = async () => {
410     ...
411   };
412
413 >   // Delete note
414 >   const deleteNote = async (data) => {
415     ...
416   };
417
418 >   // Search note
419 >   const onSearchNote = async (query) => {
420     ...
421   };
422
423 >   // Pinned note
424 >   const updateIsPinned = async (noteData) => {
425     ...
426   };
427
428 >   // Clear search
429 >   const handleClearSearch = () => {
430     ...
431   };
432
433 >   // Effect
434 >   useEffect(() => {
435     ...
436   }, []);
437
438 >   // Get user info
439 >   const getUserInfo = async () => {
440     ...
441   };
442
443 >   // Get all notes
444 >   const getAllNotes = async () => {
445     ...
446   };
447
448 >   // Delete note
449 >   const deleteNote = async (data) => {
450     ...
451   };
452
453 >   // Search note
454 >   const onSearchNote = async (query) => {
455     ...
456   };
457
458 >   // Pinned note
459 >   const updateIsPinned = async (noteData) => {
460     ...
461   };
462
463 >   // Clear search
464 >   const handleClearSearch = () => {
465     ...
466   };
467
468 >   // Effect
469 >   useEffect(() => {
470     ...
471   }, []);
472
473 >   // Get user info
474 >   const getUserInfo = async () => {
475     ...
476   };
477
478 >   // Get all notes
479 >   const getAllNotes = async () => {
480     ...
481   };
482
483 >   // Delete note
484 >   const deleteNote = async (data) => {
485     ...
486   };
487
488 >   // Search note
489 >   const onSearchNote = async (query) => {
490     ...
491   };
492
493 >   // Pinned note
494 >   const updateIsPinned = async (noteData) => {
495     ...
496   };
497
498 >   // Clear search
499 >   const handleClearSearch = () => {
500     ...
501   };
502
503 >   // Effect
504 >   useEffect(() => {
505     ...
506   }, []);
507
508 >   // Get user info
509 >   const getUserInfo = async () => {
510     ...
511   };
512
513 >   // Get all notes
514 >   const getAllNotes = async () => {
515     ...
516   };
517
518 >   // Delete note
519 >   const deleteNote = async (data) => {
520     ...
521   };
522
523 >   // Search note
524 >   const onSearchNote = async (query) => {
525     ...
526   };
527
528 >   // Pinned note
529 >   const updateIsPinned = async (noteData) => {
530     ...
531   };
532
533 >   // Clear search
534 >   const handleClearSearch = () => {
535     ...
536   };
537
538 >   // Effect
539 >   useEffect(() => {
540     ...
541   }, []);
542
543 >   // Get user info
544 >   const getUserInfo = async () => {
545     ...
546   };
547
548 >   // Get all notes
549 >   const getAllNotes = async () => {
550     ...
551   };
552
553 >   // Delete note
554 >   const deleteNote = async (data) => {
555     ...
556   };
557
558 >   // Search note
559 >   const onSearchNote = async (query) => {
560     ...
561   };
562
563 >   // Pinned note
564 >   const updateIsPinned = async (noteData) => {
565     ...
566   };
567
568 >   // Clear search
569 >   const handleClearSearch = () => {
570     ...
571   };
572
573 >   // Effect
574 >   useEffect(() => {
575     ...
576   }, []);
577
578 >   // Get user info
579 >   const getUserInfo = async () => {
580     ...
581   };
582
583 >   // Get all notes
584 >   const getAllNotes = async () => {
585     ...
586   };
587
588 >   // Delete note
589 >   const deleteNote = async (data) => {
590     ...
591   };
592
593 >   // Search note
594 >   const onSearchNote = async (query) => {
595     ...
596   };
597
598 >   // Pinned note
599 >   const updateIsPinned = async (noteData) => {
600     ...
601   };
602
603 >   // Clear search
604 >   const handleClearSearch = () => {
605     ...
606   };
607
608 >   // Effect
609 >   useEffect(() => {
610     ...
611   }, []);
612
613 >   // Get user info
614 >   const getUserInfo = async () => {
615     ...
616   };
617
618 >   // Get all notes
619 >   const getAllNotes = async () => {
620     ...
621   };
622
623 >   // Delete note
624 >   const deleteNote = async (data) => {
625     ...
626   };
627
628 >   // Search note
629 >   const onSearchNote = async (query) => {
630     ...
631   };
632
633 >   // Pinned note
634 >   const updateIsPinned = async (noteData) => {
635     ...
636   };
637
638 >   // Clear search
639 >   const handleClearSearch = () => {
640     ...
641   };
642
643 >   // Effect
644 >   useEffect(() => {
645     ...
646   }, []);
647
648 >   // Get user info
649 >   const getUserInfo = async () => {
650     ...
651   };
652
653 >   // Get all notes
654 >   const getAllNotes = async () => {
655     ...
656   };
657
658 >   // Delete note
659 >   const deleteNote = async (data) => {
660     ...
661   };
662
663 >   // Search note
664 >   const onSearchNote = async (query) => {
665     ...
666   };
667
668 >   // Pinned note
669 >   const updateIsPinned = async (noteData) => {
670     ...
671   };
672
673 >   // Clear search
674 >   const handleClearSearch = () => {
675     ...
676   };
677
678 >   // Effect
679 >   useEffect(() => {
680     ...
681   }, []);
682
683 >   // Get user info
684 >   const getUserInfo = async () => {
685     ...
686   };
687
688 >   // Get all notes
689 >   const getAllNotes = async () => {
690     ...
691   };
692
693 >   // Delete note
694 >   const deleteNote = async (data) => {
695     ...
696   };
697
698 >   // Search note
699 >   const onSearchNote = async (query) => {
700     ...
701   };
702
703 >   // Pinned note
704 >   const updateIsPinned = async (noteData) => {
705     ...
706   };
707
708 >   // Clear search
709 >   const handleClearSearch = () => {
710     ...
711   };
712
713 >   // Effect
714 >   useEffect(() => {
715     ...
716   }, []);
717
718 >   // Get user info
719 >   const getUserInfo = async () => {
720     ...
721   };
722
723 >   // Get all notes
724 >   const getAllNotes = async () => {
725     ...
726   };
727
728 >   // Delete note
729 >   const deleteNote = async (data) => {
730     ...
731   };
732
733 >   // Search note
734 >   const onSearchNote = async (query) => {
735     ...
736   };
737
738 >   // Pinned note
739 >   const updateIsPinned = async (noteData) => {
740     ...
741   };
742
743 >   // Clear search
744 >   const handleClearSearch = () => {
745     ...
746   };
747
748 >   // Effect
749 >   useEffect(() => {
750     ...
751   }, []);
752
753 >   // Get user info
754 >   const getUserInfo = async () => {
755     ...
756   };
757
758 >   // Get all notes
759 >   const getAllNotes = async () => {
760     ...
761   };
762
763 >   // Delete note
764 >   const deleteNote = async (data) => {
765     ...
766   };
767
768 >   // Search note
769 >   const onSearchNote = async (query) => {
770     ...
771   };
772
773 >   // Pinned note
774 >   const updateIsPinned = async (noteData) => {
775     ...
776   };
777
778 >   // Clear search
779 >   const handleClearSearch = () => {
780     ...
781   };
782
783 >   // Effect
784 >   useEffect(() => {
785     ...
786   }, []);
787
788 >   // Get user info
789 >   const getUserInfo = async () => {
790     ...
791   };
792
793 >   // Get all notes
794 >   const getAllNotes = async () => {
795     ...
796   };
797
798 >   // Delete note
799 >   const deleteNote = async (data) => {
800     ...
801   };
802
803 >   // Search note
804 >   const onSearchNote = async (query) => {
805     ...
806   };
807
808 >   // Pinned note
809 >   const updateIsPinned = async (noteData) => {
810     ...
811   };
812
813 >   // Clear search
814 >   const handleClearSearch = () => {
815     ...
816   };
817
818 >   // Effect
819 >   useEffect(() => {
820     ...
821   }, []);
822
823 >   // Get user info
824 >   const getUserInfo = async () => {
825     ...
826   };
827
828 >   // Get all notes
829 >   const getAllNotes = async () => {
830     ...
831   };
832
833 >   // Delete note
834 >   const deleteNote = async (data) => {
835     ...
836   };
837
838 >   // Search note
839 >   const onSearchNote = async (query) => {
840     ...
841   };
842
843 >   // Pinned note
844 >   const updateIsPinned = async (noteData) => {
845     ...
846   };
847
848 >   // Clear search
849 >   const handleClearSearch = () => {
850     ...
851   };
852
853 >   // Effect
854 >   useEffect(() => {
855     ...
856   }, []);
857
858 >   // Get user info
859 >   const getUserInfo = async () => {
860     ...
861   };
862
863 >   // Get all notes
864 >   const getAllNotes = async () => {
865     ...
866   };
867
868 >   // Delete note
869 >   const deleteNote = async (data) => {
870     ...
871   };
872
873 >   // Search note
874 >   const onSearchNote = async (query) => {
875     ...
876   };
877
878 >   // Pinned note
879 >   const updateIsPinned = async (noteData) => {
880     ...
881   };
882
883 >   // Clear search
884 >   const handleClearSearch = () => {
885     ...
886   };
887
888 >   // Effect
889 >   useEffect(() => {
890     ...
891   }, []);
892
893 >   // Get user info
894 >   const getUserInfo = async () => {
895     ...
896   };
897
898 >   // Get all notes
899 >   const getAllNotes = async () => {
900     ...
901   };
902
903 >   // Delete note
904 >   const deleteNote = async (data) => {
905     ...
906   };
907
908 >   // Search note
909 >   const onSearchNote = async (query) => {
910     ...
911   };
912
913 >   // Pinned note
914 >   const updateIsPinned = async (noteData) => {
915     ...
916   };
917
918 >   // Clear search
919 >   const handleClearSearch = () => {
920     ...
921   };
922
923 >   // Effect
924 >   useEffect(() => {
925     ...
926   }, []);
927
928 >   // Get user info
929 >   const getUserInfo = async () => {
930     ...
931   };
932
933 >   // Get all notes
934 >   const getAllNotes = async () => {
935     ...
936   };
937
938 >   // Delete note
939 >   const deleteNote = async (data) => {
940     ...
941   };
942
943 >   // Search note
944 >   const onSearchNote = async (query) => {
945     ...
946   };
947
948 >   // Pinned note
949 >   const updateIsPinned = async (noteData) => {
950     ...
951   };
952
953 >   // Clear search
954 >   const handleClearSearch = () => {
955     ...
956   };
957
958 >   // Effect
959 >   useEffect(() => {
960     ...
961   }, []);
962
963 >   // Get user info
964 >   const getUserInfo = async () => {
965     ...
966   };
967
968 >   // Get all notes
969 >   const getAllNotes = async () => {
970     ...
971   };
972
973 >   // Delete note
974 >   const deleteNote = async (data) => {
975     ...
976   };
977
978 >   // Search note
979 >   const onSearchNote = async (query) => {
980     ...
981   };
982
983 >   // Pinned note
984 >   const updateIsPinned = async (noteData) => {
985     ...
986   };
987
988 >   // Clear search
989 >   const handleClearSearch = () => {
990     ...
991   };
992
993 >   // Effect
994 >   useEffect(() => {
995     ...
996   }, []);
997
998 >   // Get user info
999 >   const getUserInfo = async () => {
1000    ...
1001  };
1002
1003 >   // Get all notes
1004 >   const getAllNotes = async () => {
1005    ...
1006  };
1007
1008 >   // Delete note
1009 >   const deleteNote = async (data) => {
1010    ...
1011  };
1012
1013 >   // Search note
1014 >   const onSearchNote = async (query) => {
1015    ...
1016  };
1017
1018 >   // Pinned note
1019 >   const updateIsPinned = async (noteData) => {
1020    ...
1021  };
1022
1023 >   // Clear search
1024 >   const handleClearSearch = () => {
1025    ...
1026  };
1027
1028 >   // Effect
1029 >   useEffect(() => {
1030    ...
1031  }, []);
1032
1033 >   // Get user info
1034 >   const getUserInfo = async () => {
1035    ...
1036  };
1037
1038 >   // Get all notes
1039 >   const getAllNotes = async () => {
1040    ...
1041  };
1042
1043 >   // Delete note
1044 >   const deleteNote = async (data) => {
1045    ...
1046  };
1047
1048 >   // Search note
1049 >   const onSearchNote = async (query) => {
1050    ...
1051  };
1052
1053 >   // Pinned note
1054 >   const updateIsPinned = async (noteData) => {
1055    ...
1056  };
1057
1058 >   // Clear search
1059 >   const handleClearSearch = () => {
1060    ...
1061  };
1062
1063 >   // Effect
1064 >   useEffect(() => {
1065    ...
1066  }, []);
1067
1068 >   // Get user info
1069 >   const getUserInfo = async () => {
1070    ...
1071  };
1072
1073 >   // Get all notes
1074 >   const getAllNotes = async () => {
1075    ...
1076  };
1077
1078 >   // Delete note
1079 >   const deleteNote = async (data) => {
1080    ...
1081  };
1082
1083 >   // Search note
1084 >   const onSearchNote = async (query) => {
1085    ...
1086  };
1087
1088 >   // Pinned note
1089 >   const updateIsPinned = async (noteData) => {
1090    ...
1091  };
1092
1093 >   // Clear search
1094 >   const handleClearSearch = () => {
1095    ...
1096  };
1097
1098 >   // Effect
1099 >   useEffect(() => {
1100    ...
1101  }, []);
1102
1103 >   // Get user info
1104 >   const getUserInfo = async () => {
1105    ...
1106  };
1107
1108 >   // Get all notes
1109 >   const getAllNotes = async () => {
1110    ...
1111  };
1112
1113 >   // Delete note
1114 >   const deleteNote = async (data) => {
1115    ...
1116  };
1117
1118 >   // Search note
1119 >   const onSearchNote = async (query) => {
1120    ...
1121  };
1122
1123 >   // Pinned note
1124 >   const updateIsPinned = async (noteData) => {
1125    ...
1126  };
1127
1128 >   // Clear search
1129 >   const handleClearSearch = () => {
1130    ...
1131  };
1132
1133 >   // Effect
1134 >   useEffect(() => {
1135    ...
1136  }, []);
1137
1138 >   // Get user info
1139 >   const getUserInfo = async () => {
1140    ...
1141  };
1142
1143 >   // Get all notes
1144 >   const getAllNotes = async () => {
1145    ...
1146  };
1147
1148 >   // Delete note
1149 >   const deleteNote = async (data) => {
1150    ...
1151  };
1152
1153 >   // Search note
1154 >   const onSearchNote = async (query) => {
1155    ...
1156  };
1157
1158 >   // Pinned note
1159 >   const updateIsPinned = async (noteData) => {
1160    ...
1161  };
1162
1163 >   // Clear search
1164 >   const handleClearSearch = () => {
1165    ...
1166  };
1167
1168 >   // Effect
1169 >   useEffect(() => {
1170    ...
1171  }, []);
1172
1173 >   // Get user info
1174 >   const getUserInfo = async () => {
1175    ...
1176  };
1177
1178 >   // Get all notes
1179 >   const getAllNotes = async () => {
1180    ...
1181  };
1182
1183 >   // Delete note
1184 >   const deleteNote = async (data) => {
1185    ...
1186  };
1187
1188 >   // Search note
1189 >   const onSearchNote = async (query) => {
1190    ...
1191  };
1192
1193 >   // Pinned note
1194 >   const updateIsPinned = async (noteData) => {
1195    ...
1196  };
1197
1198 >   // Clear search
1199 >   const handleClearSearch = () => {
1200    ...
1201  };
1202
1203 >   // Effect
1204 >   useEffect(() => {
1205    ...
1206  }, []);
1207
1208 >   // Get user info
1209 >   const getUserInfo = async () => {
1210    ...
1211  };
1212
1213 >   // Get all notes
1214 >   const getAllNotes = async () => {
1215    ...
1216  };
1217
1218 >   // Delete note
1219 >   const deleteNote = async (data) => {
1220    ...
1221  };
1222
1223 >   // Search note
1224 >   const onSearchNote = async (query) => {
1225    ...
1226  };
1227
1228 >   // Pinned note
1229 >   const updateIsPinned = async (noteData) => {
1230    ...
1231  };
1232
1233 >   // Clear search
1234 >   const handleClearSearch = () => {
1235    ...
1236  };
1237
1238 >   // Effect
1239 >   useEffect(() => {
1240    ...
1241  }, []);
1242
1243 >   // Get user info
1244 >   const getUserInfo = async () => {
1245    ...
1246  };
1247
1248 >   // Get all notes
1249 >   const getAllNotes = async () => {
1250    ...
1251  };
1252
1253 >   // Delete note
1254 >   const deleteNote = async (data) => {
1255    ...
1256  };
1257
1258 >   // Search note
1259 >   const onSearchNote = async (query) => {
1260    ...
1261  };
1262
1263 >   // Pinned note
1264 >   const updateIsPinned = async (noteData) => {
1265    ...
1266  };
1267
1268 >   // Clear search
1269 >   const handleClearSearch = () => {
1270    ...
1271  };
1272
1273 >   // Effect
1274 >   useEffect(() => {
1275    ...
1276  }, []);
1277
1278 >   // Get user info
1279 >   const getUserInfo = async () => {
1280    ...
1281  };
1282
1283 >   // Get all notes
1284 >   const getAllNotes = async () => {
1285    ...
1286  };
1287
1288 >   // Delete note
1289 >   const deleteNote = async (data) => {
1290    ...
1291  };
1292
1293 >   // Search note
1294 >   const onSearchNote = async (query) => {
1295    ...
1296  };
1297
1298 >   // Pinned note
1299 >   const updateIsPinned = async (noteData) => {
1300    ...
1301  };
1302
1303 >   // Clear search
1304 >   const handleClearSearch = () => {
1305    ...
1306  };
1307
1308 >   // Effect
1309 >   useEffect(() => {
1310    ...
1311  }, []);
1312
1313 >   // Get user info
1314 >   const getUserInfo = async () => {
1315    ...
1316  };
1317
1318 >   // Get all notes
1319 >   const getAllNotes = async () => {
1320    ...
1321  };
1322
1323 >   // Delete note
1324 >   const deleteNote = async (data) => {
1325    ...
1326  };
1327
1328 >   // Search note
1329 >   const onSearchNote = async (query) => {
1330    ...
1331  };
1332
1333 >   // Pinned note
1334 >   const updateIsPinned = async (noteData) => {
1335    ...
1336  };
1337
1338 >   // Clear search
1339 >   const handleClearSearch = () => {
1340    ...
1341  };
1342
1343 >   // Effect
1344 >   useEffect(() => {
1345    ...
1346  }, []);
1347
1348 >   // Get user info
1349 >   const getUserInfo = async () => {
1350    ...
1351  };
1352
1353 >   // Get all notes
1354 >   const getAllNotes = async () => {
1355    ...
1356  };
1357
1358 >   // Delete note
1359 >   const deleteNote = async (data) => {
1360    ...
1361  };
1362
1363 >   // Search note
1364 >   const onSearchNote = async (query) => {
1365    ...
1366  };
1367
1368 >   // Pinned note
1369 >   const updateIsPinned = async (noteData) => {
1370    ...
1371  };
1372
1373 >   // Clear search
1374 >   const handleClearSearch = () => {
1375    ...
1376  };
1377
1378 >   // Effect
1379 >   useEffect(() => {
1380    ...
1381  }, []);
1382
1383 >   // Get user info
1384 >   const getUserInfo = async () => {
1385    ...
1386  };
1387
1388 >   // Get all notes
1389 >   const getAllNotes = async () => {
1390    ...
1391  };
1392
1393 >   // Delete note
1394 >   const deleteNote = async (data) => {
1395    ...
1396  };
1397
1398 >   // Search note
1399 >   const onSearchNote = async (query) =>
```



2.2 Backend Teknolojileri

- Node.js: Sunucu tarafı JavaScript çalışma ortamı
- Express.js: Web uygulaması ve API geliştirme framework'ü
- Mongoose: MongoDB ile etkileşim için Object Data Modeling (ODM) kütüphanesi
- JSON Web Token (JWT): Güvenli kullanıcı kimlik doğrulama mekanizması
- dotenv: Ortam değişkenlerini yönetme
- cors: Cross-Origin Resource Sharing yönetimi

Backend geliştirme süreci, güvenli ve ölçülebilir bir API mimarisi oluşturmaya odaklandı. Node.js ve Express.js kombinasyonu, hızlı ve esnek bir sunucu tarafı çözümü sundu. Mongoose ORM kullanılarak MongoDB ile olan veri etkileşimi optimize edildi ve veri modellemesi için güçlü bir altyapı kuruldu.

Kullanıcı ve not modellerinin tasarımında, veri bütünlüğü ve güvenliği ön planda tutuldu. JWT (JSON Web Token) ile kimlik doğrulama mekanizması uygulandı, bu sayede kullanıcı oturumları güvenli bir şekilde yönetildi. Her API endpoint'i, hem güvenlik hem de performans açısından detaylı olarak test edildi.

```
backend > index.js > authenticateToken
1  require("dotenv").config();
2
3  const config = require("./config.json");
4  const mongoose = require("mongoose");
5
6  mongoose.connect(config.connectionstring);
7
8  const User = require("./models/user.model");
9  const Note = require("./models/note.model");
10
11 const express = require("express");
12 const cors = require("cors");
13 const app = express();
14
15 const jwt = require("jsonwebtoken");
16 const { authenticateToken, authenticateToken } = require("./utilities");
17
18 app.use(express.json());
19
20 app.use(
21   cors({
22     origin: "*",
23   })
24 );
25
26
27 // Create Account
28 > app.post("/create-account", async (req, res) => {
29   ...
30 });
31
32 // Login
33 > app.post("/login", async (req, res) => {
34   ...
35 });
36
37 // Get User
38 > app.get("/get-user", authenticateToken, async (req, res) => {
39   ...
40 });
41
42 // Add Note
43 > app.post("/add-note", authenticateToken, async (req, res) => {
44   ...
45 });
46
47 // Edit Note
48 > app.put("/edit-note/:noteId", authenticateToken, async (req, res) => {
49   ...
50 });
51
52 // Delete Note
53 > app.delete("/delete-note/:noteId", authenticateToken, async (req, res) => {
54   ...
55 });
56
```

```

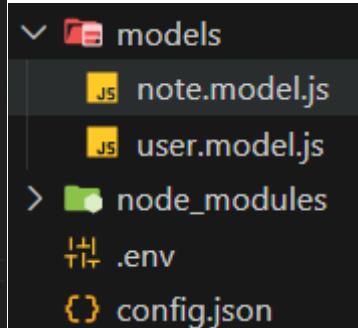
253 // Get All Notes
254 > app.get("/get-all-notes", authenticateToken, async (req, res) => { ...
271 });
272
273 // Update isPinned Value
274 > app.put("/update-note-pinned/:noteId", authenticateToken, async (req, res) => { ...
301 });
302
303 // Search notes
304 > app.get("/search-notes/", authenticateToken, async (req, res) => { ...
334 });
335
336 app.listen(8000);
337
338 module.exports = app;

```

```

backend > ls utilities.js > [o] <unknown> > authenticateToken
1 const jwt = require("jsonwebtoken");
2
3 function authenticateToken(req, res, next) {
4   const authHeader = req.headers["authorization"];
5   const token = authHeader && authHeader.split(" ")[1];
6
7   if (!token) return res.sendStatus(401);
8
9   jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, user) => {
10     if (err) return res.sendStatus(401);
11     req.user = user;
12     next();
13   });
14 }
15
16 module.exports = [
17   authenticateToken,
18 ];
19

```



2.3 Veritabanı

- MongoDB: NoSQL, esnek ve ölçülebilir bulut tabanlı veritabanı

```

backend > {} config.json > ...
1 {
2   "connectionString": "mongodb+srv://borak:TIfrzPJbBMw6krxq@skynotes.gd8
3 }

```

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar lists the project (Project 0), Data Services, Overview, DATABASE (Clusters), SERVICES (Atlas Search, Stream Processing, Triggers, Migration, Data Federation), SECURITY (Quickstart, Backup, Database Access, Network Access, Advanced), and Goto. In the main area, under the 'test' database, there is a 'users' collection. The 'Find' tab is selected, showing two documents:

```

{
  "_id": "681fbccf95a73db72894ble5",
  "fullName": "Bora",
  "email": "kullanici@posta.com",
  "password": "güçlüparola",
  "createdAt": "2025-05-10T20:50:28.532+00:00",
  "__v": 0
}

{
  "_id": "68208c5115fb0439c48d9ff7f",
  "fullName": "Bora",
  "email": "aob.co",
  "password": "1",
  "createdAt": "2025-05-11T11:34:18.132+00:00",
  "__v": 0
}

```

3. Uygulama Özellikleri

3.1 Kullanıcı İşlevselligi

- Kullanıcı kayıt ve giriş sistemi
- Not oluşturma, düzenleme ve silme
- Not arama fonksiyonu
- Notları sabitleme (pin) özelliği
- Okunabilir tarih formatı
- Bilgilendirme mesajları (Toast)
- Boş not listesi için özel kullanıcı arayüzü

4. Geliştirme Süreci

4.1 Proje Kurulumu

- Proje npm create vite@latest kullanılarak oluşturuldu
- Tailwind CSS resmi dokümantasyonuna göre entegre edildi
- Google Fonts'dan Noto Sans fontu kullanıldı
- Tarih biçimlendirme için moment kütüphanesi entegre edildi

4.2 Frontend Geliştirme Aşamaları

- Sayfa ve bileşen tasarımlı
- React Router DOM ile navigasyon
- Tailwind CSS ile stil oluşturma
- Özel bileşenler geliştirildi:
 - Navbar
 - Giriş ekranı
 - Şifre girişi bileşeni
 - Arama çubuğu
 - Not ekleme/düzenleme modali
 - Etiket girişi bileşeni

4.3 Backend Geliştirme Aşamaları

- Node.js, Express.js ve Mongoose ile API geliştirme
- Kullanıcı ve Not modelleri oluşturma
- API endpoint'leri hazırlama:
 - Kullanıcı kayıt
 - Kullanıcı girişi

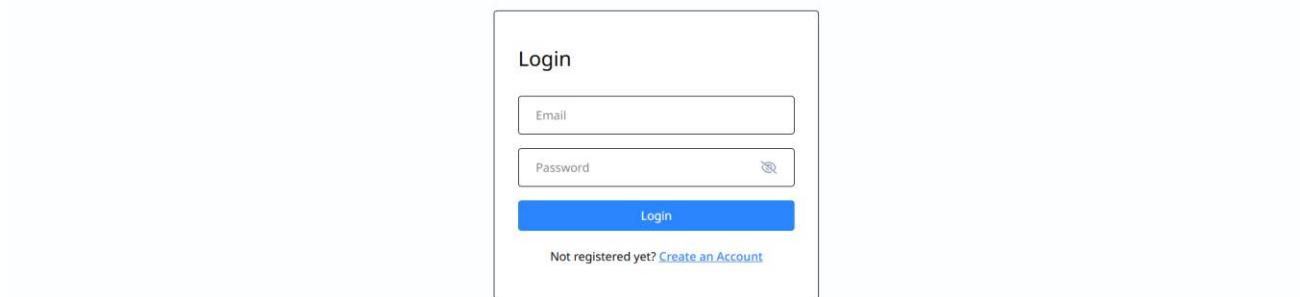
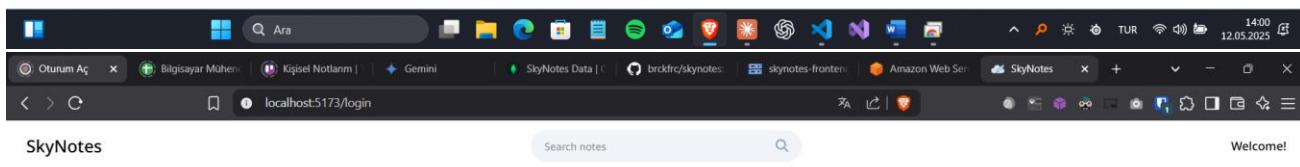
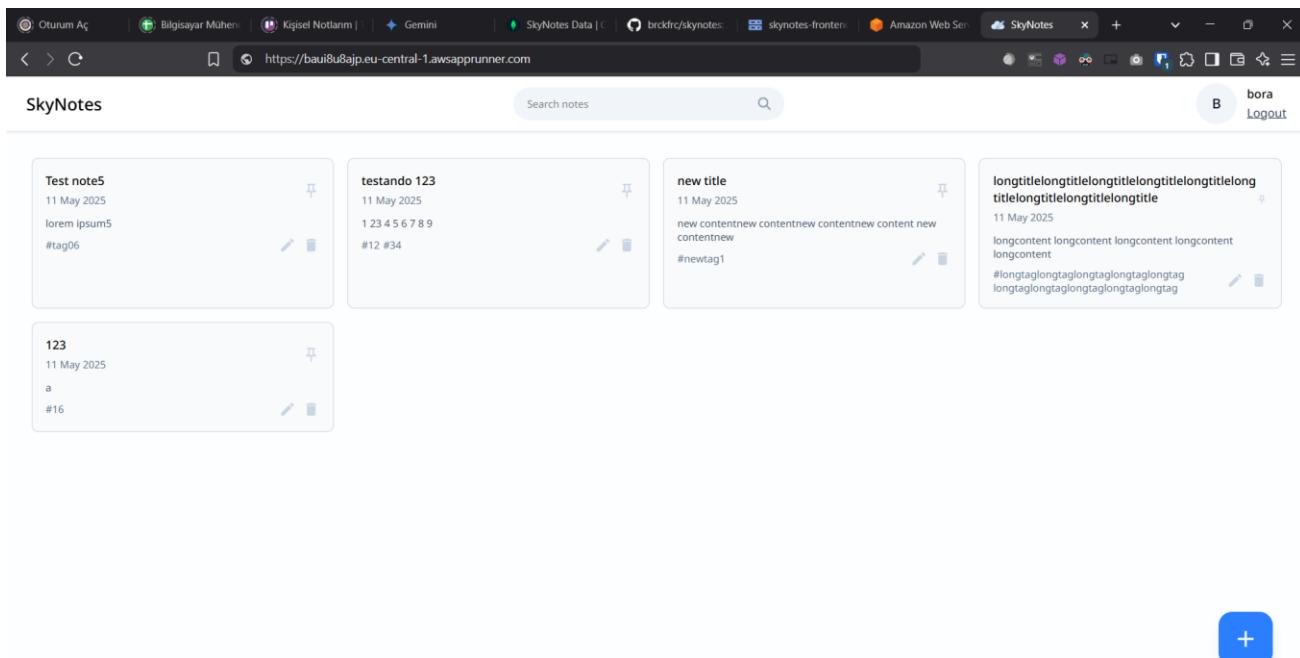
- Not oluşturma
- Not düzenleme
- Not silme
- Not getirme
- Not sabitleme

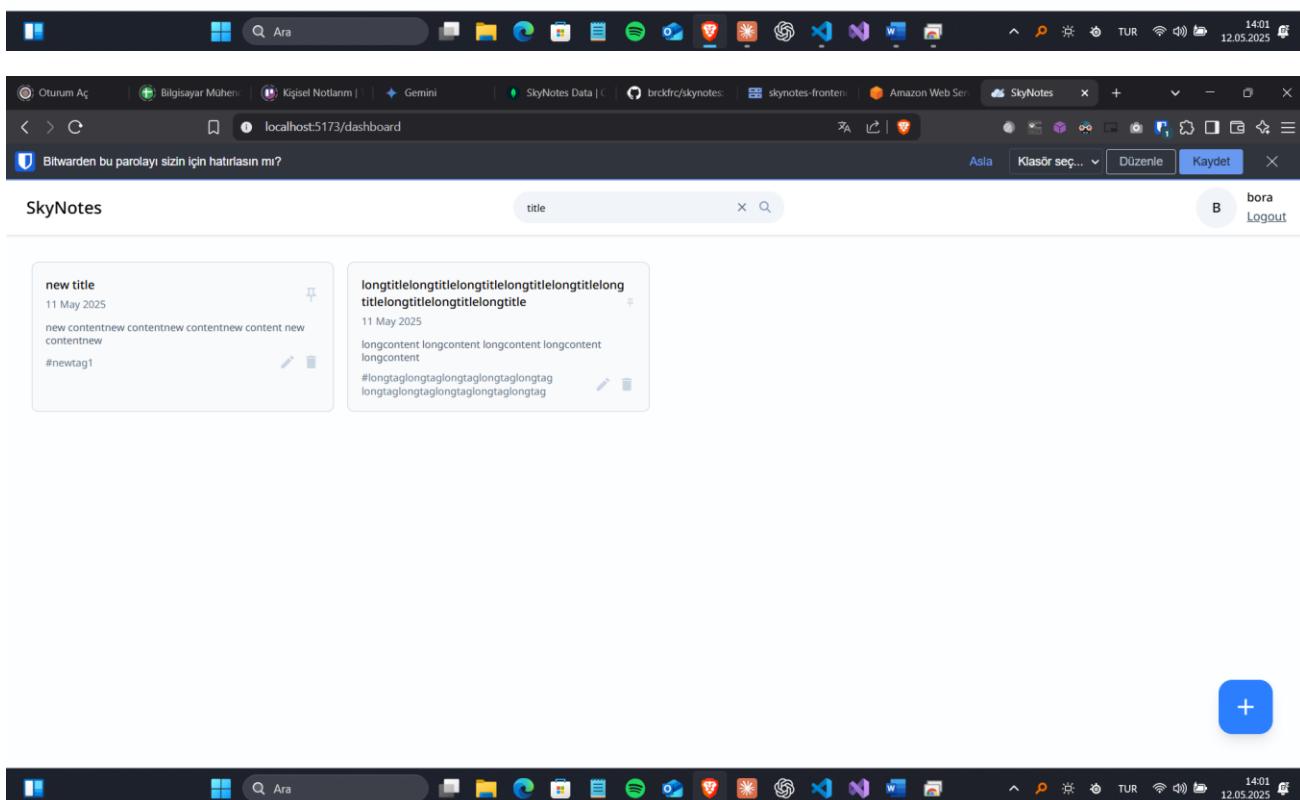
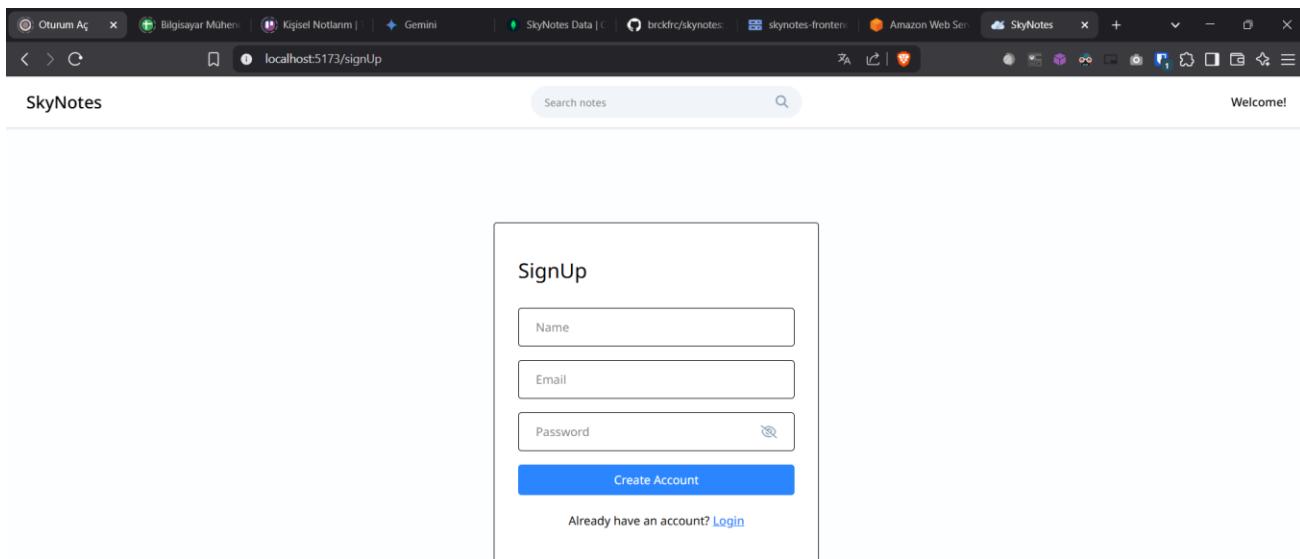
4.4 Entegrasyon

- Frontend ve backend arasında API bağlantıları kuruldu
- Kullanıcı bilgileri backend'den çekilip görüntüülendi
- Kullanıcıya özel notlar listelendi
- Arama fonksiyonu entegre edildi
- Bilgilendirme mesajları (toast) eklendi

5. Sonuç

SkyNotes projesi, modern web geliştirme teknolojilerini kullanarak kullanıcı merkezli, esnek bir not yönetim uygulaması oluşturmayı başarmıştır. Proje, frontend ve backend teknolojilerinin entegrasyonunu gösteren kapsamlı bir örnek teşkil etmektedir.





PROJE 2: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

Proje Konusu: AWS kullanarak telefon özelliklerini bulunduran veri setini eğiterek fiyat aralık tahmini yapan modeli buluta deploy ettik ve buradan API oluşturarak veritabanı uygulamasında veri analizi gerçekleştirdik.

Bu projede;

Backend dili: Python, Node.js

Makine Öğrenmesi Kütüphaneleri,

Veri Tabanı: MongoDB,

Bulut Platformu: AWS (S3 Bucket, Sagemaker, Lambda, API Gateway)

code

Generate + Code + Markdown

```
import sagemaker
from sklearn.model_selection import train_test_split
import boto3
import pandas as pd

sm_boto3 = boto3.client("sagemaker", region_name="us-east-1")
sess = sagemaker.Session(boto_session=boto3.Session(region_name="us-east-1"))
region = sess.boto_session.region_name
bucket = 'mlbucketpricemobile'
print("Using bucket"+ bucket)
```

S3 Bucket oluşturarak localdeki kod ile bağlantı sağlandı.

```
● df.isnull().mean() * 100  
  
battery_power      0.0  
blue              0.0  
clock_speed       0.0  
dual_sim          0.0  
fc                0.0  
four_g            0.0  
int_memory        0.0  
m_dep             0.0  
mobile_wt         0.0  
n_cores           0.0  
pc                0.0  
px_height         0.0  
px_width          0.0  
ram               0.0  
sc_h              0.0  
sc_w              0.0  
talk_time         0.0  
three_g           0.0  
touch_screen      0.0  
wifi              0.0
```

Elimizde bulunan dataset'i analiz edilerek eksiklikler gözlemlendi.

```
trainX = pd.DataFrame(X_train)
trainX[label] = y_train

testX = pd.DataFrame(X_test)
testX[label] = y_test
```

Python

```
trainX.to_csv("train-V-1.csv", index=False )
testX.to_csv("test-V-1.csv", index=False )

sk_prefix = "sagemaker/mobile_price_classification/sklearncontainer"
trainpath = sess.upload_data(
    path ="train-V-1.csv", bucket=bucket, key_prefix=sk_prefix
)

testpath = sess.upload_data(
    path ="test-V-1.csv", bucket=bucket, key_prefix=sk_prefix
)
print(trainpath)
print(testpath)
```

Python

Python

Bucket' a var olan train.csv ile test.csv dosyaları yüklandı.

```
1
2  from sklearn.ensemble import RandomForestClassifier
3  from sklearn.metrics import accuracy_score, classification_report
4  import pandas as pd
5  import numpy as np
6  import joblib
7  import os
8  import json
9  from io import StringIO
10 import argparse
11
12
13 def model_fn(model_dir):
14     clf = joblib.load(os.path.join(model_dir, "model.joblib"))
15     return clf
16
17
18 def input_fn(request_body, request_content_type):
19
20     if request_content_type == "application/json":
21         data = json.loads(request_body)
22         return pd.DataFrame(data)
23     else:
24         raise ValueError(f"Desteklenmeyen content type: {request_content_type}")
25
26 def output_fn(prediction, content_type):
27
28     if content_type == "application/json":
29         return json.dumps({"prediction": prediction.tolist()})
30     else:
31         raise ValueError(f"Desteklenmeyen content type: {content_type}")
32
33 if __name__ == "__main__":
34
35     parser = argparse.ArgumentParser()
36     parser.add_argument("--n_estimators", type=int, default=100)
37     parser.add_argument("--random_state", type=int, default=0)
38     parser.add_argument("--max_depth", type=int, default=None)
39
40     parser.add_argument("--model-dir", type=str, default=os.environ.get("SM_MODEL_DIR"))
41     parser.add_argument("--train", type=str, default=os.environ.get("SM_CHANNEL_TRAIN"))
42     parser.add_argument("--test", type=str, default=os.environ.get("SM_CHANNEL_TEST"))
43
44
45     parser.add_argument("--train-file", type=str, default="train-V-1.csv")
46     parser.add_argument("--test-file", type=str, default="test-V-1.csv")
```



```

X_train = train_df[features]
X_test = test_df[features]
y_train = train_df[label]
y_test = test_df[label]

model = RandomForestClassifier(
    n_estimators=args.n_estimators,
    random_state=args.random_state,
    max_depth=args.max_depth
)
model.fit(X_train, y_train)

model_path = os.path.join(args.model_dir, "model.joblib")
joblib.dump(model, model_path)

y_pred = model.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

```

Randomforest ile elimizde bulunan dataseti eğitmek için ve daha sonra model eğitiminde kullanılması için script.py üzerine overwrite edildi.

- ```

from sagemaker.sklearn.model import SKLearnModel
from time import gmtime, strftime

model_name = "Custom-sklearn-model-" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
model = SKLearnModel(
 model_data=artifact,
 role="arn:aws:iam::767141477485:role/MLsagemaker",
 entry_point="script.py",
 framework_version=FRAMEWORK_VERSION,
 sagemaker_session=sagemaker_session
)
```

Aws üzerinde IAM rolü belirlendi ve bağlantı kurması için koda eklendi. Arından script.py ile model eğitildi.

```

● from time import gmtime, strftime
import time

endpoint_name = f"mobile-price-classifier-{int(time.time())}"
print("EndpointName={}".format(endpoint_name))

predictor = model.deploy(
 initial_instance_count=1,
 instance_type="ml.m4.xlarge",
 endpoint_name=endpoint_name,
 sagemaker_session=sagemaker_session
)

```

Bu kısımda Sagemaker için endpoint oluşturuldu.

**Roller (23) Bilgi**

IAM rolü, kısa süreler için geçerli olan kimlik bilgilerine ve belirli izinlere sahip olacak şekilde oluşturabileceğiniz bir kimliktir. Roller, güvenliğiniz varlıklar tarafından üstlenilebilir.

| Rol adı                           | Güvenilir varlıklar    | Son etkinlik  |
|-----------------------------------|------------------------|---------------|
| ml_price_calculator-role-dg0rchny | AWS Hizmeti: lambda    | 8 gün önce    |
| MLSagemaker                       | AWS Hizmeti: sagemaker | 6 dakika önce |

**ml\_price\_calculator**

**İşlev genel bakış Bilgi**

Diyagram | Şablon

**Açıklama**

Son değiştirilme  
2 hafta önce

İşlev ARN'si  
arnaws:lambda:us-east-1:767141477485:function:ml\_price\_calculator

İşlev URL'si | Bilgi

**Kod kaynağı Bilgi**

Şuradan yükle ▾

Ardından, Lambdaya IAM üzerinden sagemaker erişim rolü verildi. Ardından bu endpoint Lambda'ya bağlandı.

The screenshot shows the AWS API Gateway interface. On the left, a sidebar lists various API-related settings like APIs, Stages, Authorizers, and API keys. The main area is titled 'Resources' and shows a single resource path: '/api-ml-price-calculator' with a 'POST' method. The 'Resource details' section shows the path as '/' and the Resource ID as 'w38hftlink'. The 'Methods (0)' section indicates 'No methods' defined. There are buttons for 'Update documentation', 'Deploy API', 'Delete', and 'Create method'.

API Gateway kullanılarak MongoDB için RESTAPI oluşturuldu ve deploy edildi

A terminal window displays a Python script named 'mongo.py'. The script imports 'requests' and 'pymongo', and defines a URL and headers for a POST request to the '/api-ml-price-calculator' endpoint. It then constructs a payload dictionary containing various device specifications and sends the request. The right side of the screen shows a blurred screenshot of the AWS Lambda function configuration page.

```
mongo.py > ...
1 import requests
2 from pymongo import MongoClient
3 import json
4
5
6 url = "https://1hfgp7cu95.execute-api.us-east-1.amazonaws.com/prod/api-ml-price-calculator"
7 headers = {"Content-Type": "application/json"}
8
9
10 payload = {
11 "battery_power": 1454,
12 "blue": 1,
13 "clock_speed": 0.5,
14 "dual_sim": 1,
15 "fc": 1,
16 "four_g": 0,
17 "int_memory": 34,
18 "m_dep": 0.7,
19 "mobile_wt": 83,
20 "n_cores": 4,
21 "pc": 2,
22 "px_height": 250,
23 "px_width": 1033,
24 "ram": 3419,
25 "sim": 1}
```

```
30 "wifi": 0
31 }
32
33
34 try:
35 response = requests.post(url, json=payload, headers=headers)
36 response.raise_for_status()
37
38
39 result = response.json()
40
41 body = json.loads(result["body"])
42 print("Prediction sonucu:", body)
43
44 except requests.exceptions.RequestException as e:
45 print("Veri çekme hatası:", e)
46 exit()
47
48
49 try:
50 client = MongoClient("mongodb+srv://useradmin:admin@mlprojecluster.dl7if0o.mongodb.net/?retryWrites=true&w=majority&appName=m+mlproje")
51 db = client["awsmlproje"]
52 collection = db["awsmlprojecollection"]
53 print("MongoDB'ye bağlandı.")
54 except Exception as e:
55 print("MongoDB'ye bağlanma hatası:", e)
56 exit()
57
58 try:
59 payload_with_prediction = payload.copy()
60 payload_with_prediction["prediction"] = body["prediction"][0]
61 collection.insert_one(payload_with_prediction)
62 print("Veri MongoDB'ye kaydedildi.")
63 except Exception as e:
64 print("Veri kaydetme hatası:", e)
65
```

Bu kısımda MongoDB ile AWS API bağlantısı sağlandı ardından local kodda bu bağlantı gösterildi. Test verisi için veri yollandı ve ardından kod çalıştırılarak API kullanıldı

The screenshot shows the Compass interface for a MongoDB database named 'awsmlproje' and a collection named 'awsmlprojecollection'. The top navigation bar includes 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A button for 'INSERT DOCUMENT' is located in the top right. Below the navigation is a search bar with the placeholder 'Type a query: { field: 'value' }' and buttons for 'Reset', 'Apply', and 'Options'. The main area displays the results of a query with the message 'QUERY RESULTS: 1-1 OF 1'. The document returned is:

```
_id: ObjectId('68166e41804982760795e13e')
battery_power : 1454
blue : 1
clock_speed : 0.5
dual_sim : 1
fc : 1
four_g : 0
int_memory : 34
m_dep : 0.7
mobile_wt : 83
n_cores : 4

pc :
px_height : 250
px_width : 1033
ram : 3419
sc_h : 7
sc_w : 5
talk_time : 5
three_g : 1
touch_screen : 1
wifi : 0
prediction : 3
```

Sonuç olarak, MongoDB'ye Tahmin ve test verisi gönderildi ve gözlemlendi.

## **PROJE 3: E-Ticaret Uygulaması (Otomatik Ölçeklendirme ve Yönetim)**

**Proje Konusu: Microsoft Azure ile E-Ticaret sitesi**

<https://nice-sand-04de3bb0f.6.azurestaticapps.net>

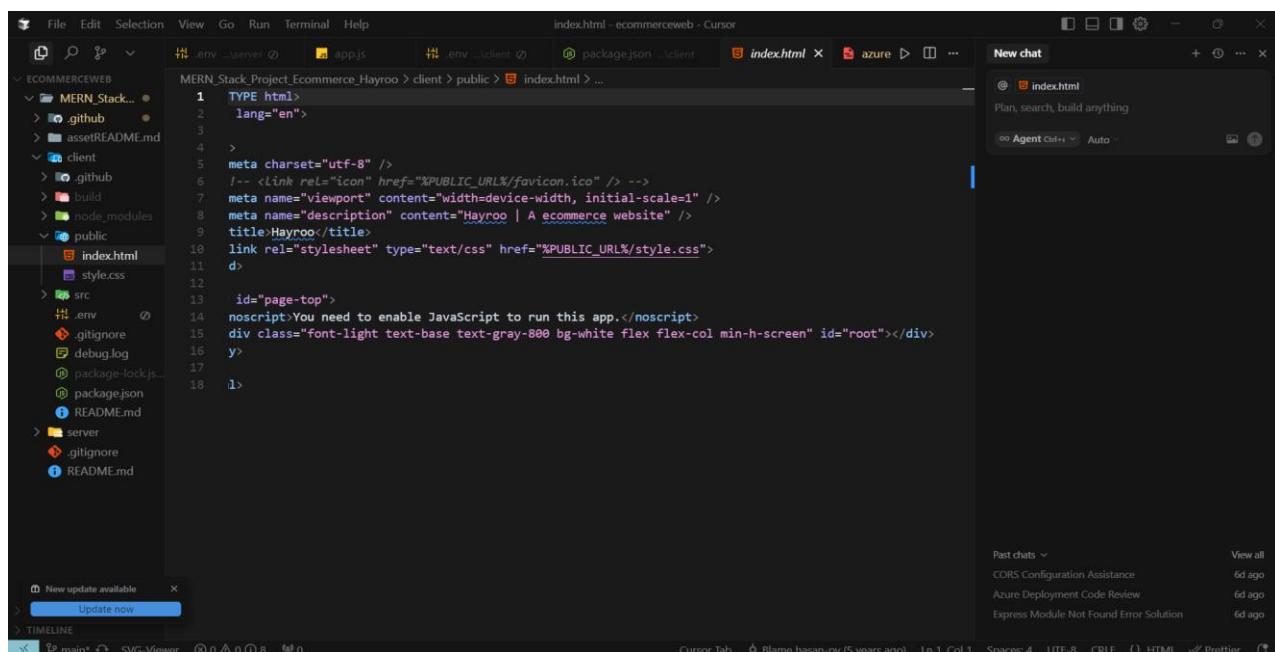
**Bu projede;**

**Frontend dili: JavaScript,**

**Backend dili: Node.js,**

**Veri Tabanı: MongoDB,**

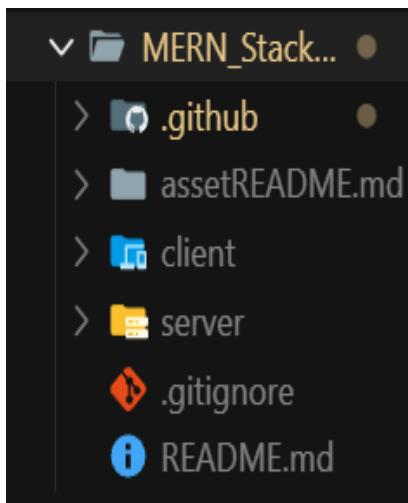
**Bulut Platformu: Microsoft Azure (Azure App Services)**



```
TYPE html>
lang="en">
>
meta charset="utf-8" />
!-- <link rel="icon" href="%PUBLIC_URL%/favicon.ico" /> -->
meta name="viewport" content="width=device-width, initial-scale=1" />
meta name="description" content="Hayroo | A ecommerce website" />
title:Hayroo</title>
link rel="stylesheet" type="text/css" href="%PUBLIC_URL%/style.css">
d>
id="page-top">
noscript>You need to enable JavaScript to run this app.</noscript>
div class="font-light text-base text-gray-800 bg-white flex flex-col min-h-screen" id="root"></div>
y>
l>
```

Bu projede, Cloud bağlantı yapılabilecek hazır website taslağı kullanıldı. Src:

[https://github.com/hasan-py/MERN\\_Stack\\_Project\\_Ecommerce\\_Hayroo](https://github.com/hasan-py/MERN_Stack_Project_Ecommerce_Hayroo)



server dosyası backendi barındırırken (node.js),  
client dosyası frontendi barındırmaktadır. (Html, Css,  
Javascript)

The screenshot shows the Microsoft Azure portal interface for the 'ecommerce-cloudproject' resource group. The left sidebar shows various navigation options like Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, Cost Management, Monitoring, Automation, and Help. The main area displays the following resources:

| Name                     | Type             | Location    | Actions |
|--------------------------|------------------|-------------|---------|
| ecommerce-frontend       | Static Web App   | West Europe | ...     |
| ecommerce-frontendgithub | Static Web App   | East US 2   | ...     |
| ecommerce-plan           | App Service plan | West Europe | ...     |
| ecommercecloud           | App Service      | West Europe | ...     |

İlk olarak Microsoft Azure üzerinde dosyaların barınacağı Resource Group oluşturuldu.

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

tinaztepe124@gmail.com VARSAYILAN DIZIN

Home > ecommerce-plan

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events (preview)

Resource visualizer

Settings

Monitoring

Automation

Help

https://portal.azure.com/#

Search

Delete Send us your feedback

App Service Plan metrics are not applicable to Free and Shared SKU

Essentials

Resource group (move) : ecommerce-cloudproject

Status : Ready

Location : West Europe

Subscription (move) : Azure subscription 1

Subscription ID : 7db76202-793c-44a4-a7c9-631894610496

Pricing plan : F1

Instance count : 1

App(s) / Slots : 1/0

Operating System : Linux

Zone redundant : Disabled

Tags (edit) : Add tags

CPU Percentage

Memory Percentage

Data In

JSON View

Devamında, Linux tabanlı plan hizmete ayrıldı ve resource group bağlantısı yapıldı.

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

tinaztepe124@gmail.com VARSAYILAN DIZIN

Home > ecommercecloud

Web App

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Events (preview)

Recommended services (preview)

Resource visualizer

Deployment

Settings

Performance

App Service plan

Download tools Add or remove favorites by pressing Ctrl+Shift+F

Search

Browse Stop Swap Restart Delete Refresh Download publish profile Reset publish profile Share to mobile ...

Essentials

Resource group (...) : ecommerce-cloudproject

Status : Running

Location (move) : West Europe

Subscription (move) : Azure subscription 1

Subscription ID : 7db76202-793c-44a4-a7c9-631894610496

Default domain : ecommercecloud.azurewebsites.net

App Service Plan : ecommerce-plan (F1: 1)

Operating System : Linux

Health Check : Not Configured

Tags (edit) : Add tags

Properties Monitoring Logs Capabilities Notifications Recommendations

Web app

Name : ecommercecloud

Publishing model : Code

Runtime Stack : Node - 20-Its

Domains

JSON View

Bu kısımda, Azure Web App Service oluşturuldu.

ecommercecloud.azurewebsites.net

okunaklı hale getir □

{"message": "E-commerce API is running"}

Azure Web App Service üzerinde localdeki kodun backend kısmı ile App Service API ile bağlı. Backend sunucuya yüklendi.

Microsoft Azure   [Upgrade](#)      [Copilot](#)        [tinaztepe124@gmail.com](#)   [VARSAYILAN DİZİN](#)

Home >

# ecommerce-frontendgithub

Static Web App

  [View app in browser](#)    Refresh    Delete    Manage deployment token    Send us your feedback

[Overview](#)   [JSON View](#)

[Activity log](#)

[Access control \(IAM\)](#)

[Tags](#)

[Diagnose and solve problems](#)

[Resource visualizer](#)

[Settings](#)

[Monitoring](#)

[Automation](#)

[Help](#)

**Essentials**

|                     |                                          |                    |                                                                                                                   |
|---------------------|------------------------------------------|--------------------|-------------------------------------------------------------------------------------------------------------------|
| Resource group (... | : <a href="#">ecommerce-cloudproject</a> | URL                | : <a href="https://nice-sand-04de3bb0f6.azurestaticapps.net">https://nice-sand-04de3bb0f6.azurestaticapps.net</a> |
| Subscription (move) | : <a href="#">Azure subscription 1</a>   | Source             | : <a href="#">main (GitHub)</a>                                                                                   |
| Subscription ID     | : 7db76202-793c-44a4-a7c9-631894610496   | Deployment history | : <a href="#">GitHub Action runs</a>                                                                              |
| Location            | : Global                                 | View workflow      | : <a href="#">azure-static-web-apps-nice-sand-04de3bb0f.yml</a>                                                   |
| Sku                 | : Free                                   |                    |                                                                                                                   |
| Tags (edit)         | : <a href="#">Add tags</a>               |                    |                                                                                                                   |

[Get started](#)   [Monitoring](#)   [Deployment history](#)



**View your application**

|                                                                                         |             |                                                                                                                 |              |
|-----------------------------------------------------------------------------------------|-------------|-----------------------------------------------------------------------------------------------------------------|--------------|
| Status                                                                                  | Environment | Domain                                                                                                          | Hosting plan |
|  Ready | Production  | <a href="https://nice-sand-04de3bb0f6.azurestaticapps.net">https://nice-sand-04de3bb0f6.azurestaticapps.net</a> | Free         |

[Visit your site](#)

Microsoft Azure üzerinde Static Web App oluşturuldu. Localdeki frontend kodu Github aracılığıyla deploy edildi.

| Actions                                                                |                              |
|------------------------------------------------------------------------|------------------------------|
|                                                                        | <a href="#">New workflow</a> |
| <a href="#">All workflows</a>                                          |                              |
| Azure Static Web Apps CI/CD                                            |                              |
| Azure Static Web Apps CI/CD                                            |                              |
| Management                                                             |                              |
| <a href="#">Caches</a>                                                 |                              |
| <a href="#">Attestations</a>                                           | ↗                            |
| <a href="#">Runners</a>                                                |                              |
| <a href="#">Usage metrics</a>                                          | ↗                            |
| <a href="#">Performance metrics</a>                                    | ↗                            |
| <hr/>                                                                  |                              |
| <b>CORS configuration updated</b>                                      |                              |
| Azure Static Web Apps CI/CD #8: Commit dadfd18 pushed by hsyntinaztepe | <a href="#">main</a>         |
|                                                                        | last week<br>2m 4s<br>...    |
| <b>CORS configuration updated</b>                                      |                              |
| Azure Static Web Apps CI/CD #9: Commit dadfd18 pushed by hsyntinaztepe | <a href="#">main</a>         |
|                                                                        | last week<br>2m 7s<br>...    |
| <b>Updated API URL configuration</b>                                   |                              |
| Azure Static Web Apps CI/CD #7: Commit 527eca0 pushed by hsyntinaztepe | <a href="#">main</a>         |
|                                                                        | last week<br>2m 16s<br>...   |
| <b>Updated API URL configuration</b>                                   |                              |
| Azure Static Web Apps CI/CD #8: Commit 527eca0 pushed by hsyntinaztepe | <a href="#">main</a>         |
|                                                                        | last week<br>2m 6s<br>...    |
| <b>Update build script to use export for Linux environment</b>         |                              |
| Azure Static Web Apps CI/CD #7: Commit 7412232 pushed by hsyntinaztepe | <a href="#">main</a>         |
|                                                                        | last week<br>2m 30s<br>...   |
| <b>Update build script to use export for Linux environment</b>         |                              |
| Azure Static Web Apps CI/CD #6: Commit 7412232 pushed by hsyntinaztepe | <a href="#">main</a>         |
|                                                                        | last week<br>2m 7s<br>...    |

```

MERN_Stack_Project_Ecommerce_Hayroo > server > app.js > ...
52 |
53)
54 .catch((err) => {
55 console.log("Database Connection Error:", err);
56 console.log("Connection String:", process.env.DATABASE);
57 });
58
59 // Middleware
60 app.use(morgan("dev"));
61 app.use(cookieParser());
62 app.use(
63 cors({
64 origin: "https://nice-sand-04de3bb0f.6.azurestaticapps.net",
65 credentials: true,
66 })
67);
68 app.use(express.static("public"));
69 app.use(express.urlencoded({ extended: false }));
70 app.use(express.json());
1 REACT_APP_API_URL=https://ecommercecloud.azurewebsites.net
2

```

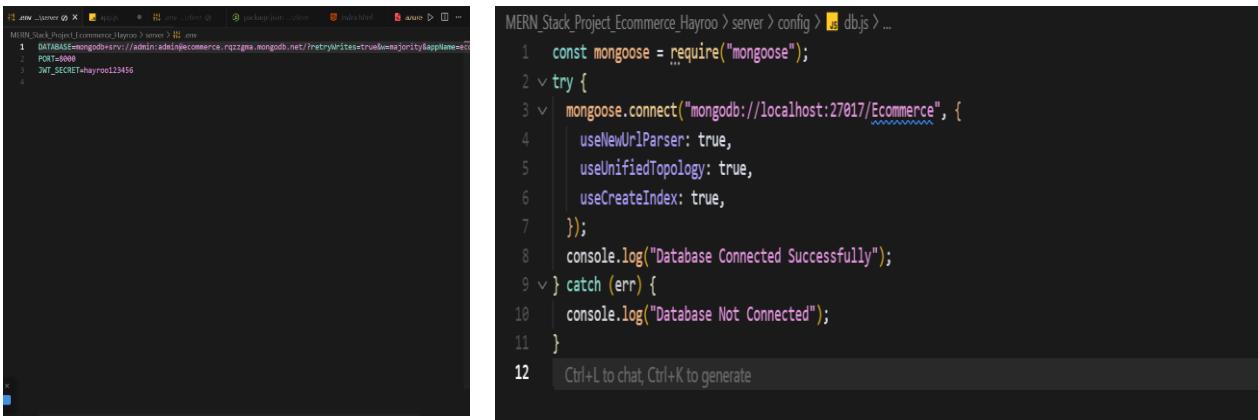
Projenin backend kısmı frontend kısmına Cloud üzerinde bağlanması için localde bağlantı sağlandı.

The screenshot shows the MongoDB Atlas Data Services interface. On the left sidebar, under the 'Clusters' section, the 'ecommerce' cluster is selected. Under 'DATABASE', the 'test' database is selected, and under it, the 'categories' collection is shown. The main panel displays the 'test.categories' collection details, including storage size (36KB), logical data size (207B), total documents (1), and index size (36KB). Below this, there are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A search bar at the top right says 'Generate queries from natural language in Compass'. At the bottom, a 'QUERY RESULTS: 1-1 OF 1' section shows a single document:

```

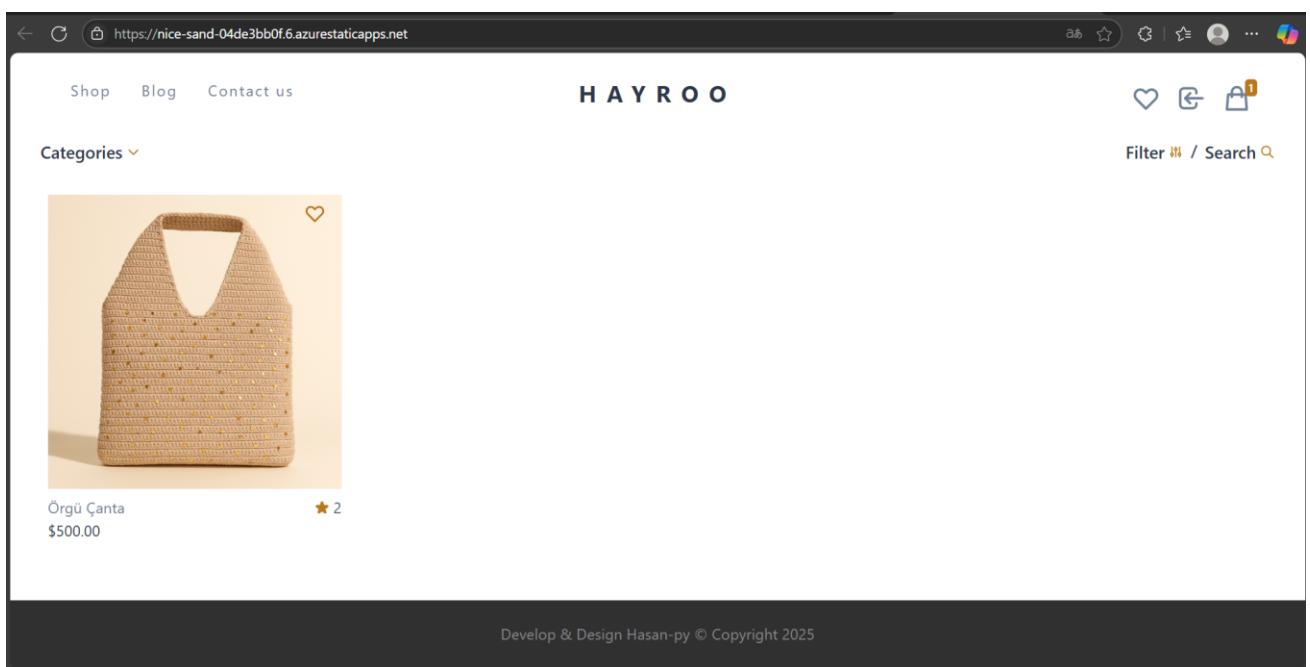
_id: ObjectId('682f28e55e1c481c383a9125')
cName : "Çanta"
cDescription : "Çanta modelleri"
cStatus : "Active"
cImage : "1747921125840_ChatGPT Image 17 May 2025 12_21_03.png"
createdAt : 2025-05-22T13:38:45.950+00:00
updatedAt : 2025-05-22T13:38:45.950+00:00
__v : 0

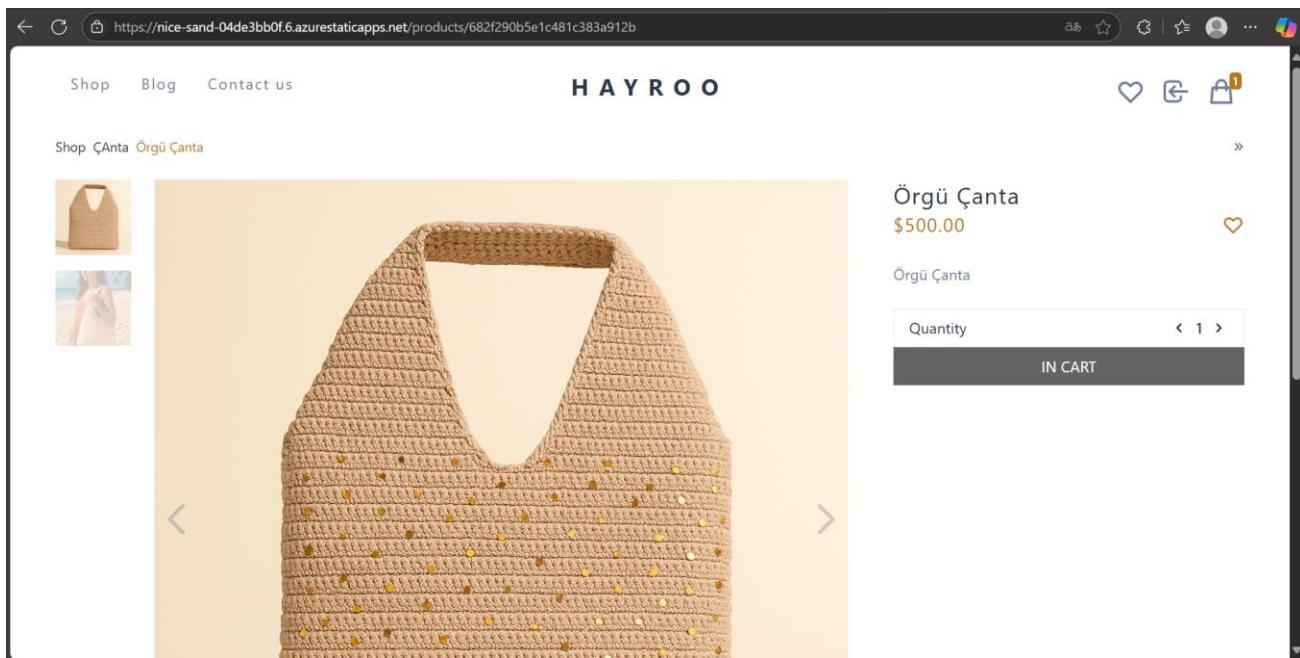
```



```
MERN_Stack_Project_Ecommerce_Hayroo > server > config > db.js > ...
1 const mongoose = require("mongoose");
2 v try {
3 v mongoose.connect("mongodb://localhost:27017/Ecommerce", {
4 useNewUrlParser: true,
5 useUnifiedTopology: true,
6 useCreateIndex: true,
7 });
8 console.log("Database Connected Successfully");
9 v } catch (err) {
10 console.log("Database Not Connected");
11 }
12 Ctrl+L to chat, Ctrl+K to generate
```

Database olarak kullanılması için MongoDB bağlantısı sağlandı.





A screenshot of the admin dashboard for the HAYROO platform. The top navigation bar includes a menu icon, the "HAYROO" logo, and search/filter icons. The left sidebar has links for "Dashboard", "Categories", "Product", and "Order". The main content area features four cards: "Customers" (7% up), "Orders" (10% up), "Product", and "Categories". Below this is a section titled "Shop Slider Images" with a "Upload File" button and a message stating "No slide image found". At the bottom, there's a section titled "Today's Orders 0" with a table header for "Products", "Image", "Status", "Order Address", and "Ordered at". A message at the bottom left says "Total 0 orders found".

Ve bulut üzerinde çalışan e-ticaret sitesi kullanıma açık.

# Sensora - IoT Sensör Verisi Gerçek Zamanlı İşleme Projesi

## 1. PROJE GENEL BAKIS

### 1.1 Proje Amacı

Bu proje, Google Cloud Platform (GCP) üzerinde gerçek zamanlı IoT sensör verilerinin toplanması, işlenmesi ve saklanması için tam entegre bir sistem geliştirmeyi amaçlamaktadır. Sistem, Cloud Functions, Pub/Sub, Firestore ve BigQuery teknolojilerini kullanarak ölçeklenebilir bir veri işleme pipeline'ı oluşturmaktadır.

### 1.2 Seçilen Proje

#### Proje 2: Gerçek Zamanlı Veri Akışı ve İşleme (IoT veya WebSocket Uygulaması)

Kullanılan teknolojiler:

- Backend Dil:** Python
- Protokol:** Pub/Sub (Google Cloud mesajlaşma servisi)
- Veritabanları:** Firestore (NoSQL), BigQuery (Veri ambarı)
- Bulut Platformu:** Google Cloud Platform

## 2. SİSTEM MİMARİSİ

### 2.1 Genel Mimari

```
[Sensor Veri Simulatoru] → [Pub/Sub Topic] → [Cloud Function] → [Firestore + BigQuery]
```

### 2.2 Bileşen Detayları

#### 1. Veri Kaynağı (Sensör Simülatörü)

- Sıcaklık ve nem verilerini simüle eder
- 5 saniye aralıklarla veri üretir
- JSON formatında Pub/Sub'a gönderir

- Asenkron mesaj kuyruğu
- Yüksek throughput desteği
- Güvenilir mesaj teslimi

#### 3. Cloud Function (process\_sensor\_data)

- Serverless işlem birimi
- Pub/Sub trigger ile otomatik tetikleme
- Python 3.13 runtime

#### 4. Veri Depolama

- Firestore:** Gerçek zamanlı erişim için
- BigQuery:** Analitik sorgular için

#### 2. Pub/Sub Topic (iot-sensor-data)

### 3. TEKNİK UYGULAMA DETAYLARI

#### 3.1 Proje Kurulumu ve Konfigürasyon

##### Google Cloud Proje Ayarları

- Proje ID: `sensora-460619`
- Bölge: `europe-west3` (Frankfurt)
- Etkinleştirilen API'ler:
  - o Cloud Functions API
  - o Pub/Sub API
  - o Firestore API
  - o BigQuery API

##### Gerekli Servisler ve Kaynaklar

```
Proje konfigürasyonu
gcloud config set project sensora-460619

Cloud Function deployment
gcloud functions deploy process_sensor_data \
 --entry-point process_sensor_data \
 --runtime python313 \
 --trigger-topic iot-sensor-data \
 --region europe-west3 \
 --project sensora-460619
```

Google Cloud projesi `sensora-460619`, Frankfurt bölgesinde (`europe-west3`) bulunuyor.

Cloud Functions, Pub/Sub, Firestore ve BigQuery API'leri etkinleştirildi.

Sağdaki görseldeki komutla proje seçilirken, `gcloud functions deploy` komutuyla Python 3.13 kullanarak `process_sensor_data` fonksiyonu, `iot-sensor-data` Pub/Sub konusu tetikleyicisiyle ilgili bölgede dağıtılmıyor.

Bu yapılandırma, sensör verilerinin bulut ortamında işlenmesini sağlıyor.

#### 3.2 Veri Akış Pipeline'i

##### Sensör Veri Simülatörü (`sensor_data_publisher.py`)

```
Ana özellikler:
- Rastgele sıcaklık verisi (20-30°C)
- Rastgele nem verisi (%40-60)
- Sabit cihaz ID (sensor-001)
- Unix timestamp ile zaman damgası
- 5 saniye aralıkları sürekli veri gönderimi
Örnek Veri Formatı:
{
 "temperature": 25.34,
 "humidity": 52.18,
 "device_id": "sensor-001",
 "timestamp": 1716590123
}
```

##### Cloud Function İşleme Mantığı (`main.py`)

```
Temel işlevler:
1. Pub/Sub mesajını decode etme
2. JSON parsing ve validasyon
3. Önceki veri ile karşılaştırma (%5 değişim kontrolü)
4. Firestore'a kaydetme
5. BigQuery'ye paralel kaydetme
6. Hata yönetimi ve loglama
```

##### Akıllı Filtreleme Algoritması:

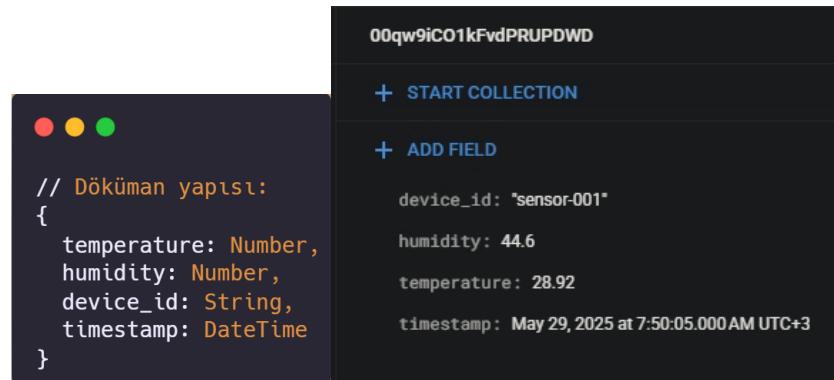
- Önceki ölçüm ile karşılaştırma
- %5'ten az değişimde veri yazılmaz
- Gereksiz veri kirliliğini önler
- Depolama maliyetini optimize eder

Sensör Veri Simülatörü, cihaz kimliği ve zaman damgasıyla birlikte 5 saniyede bir rastgele sıcaklık ve nem verisi üretip JSON formatında gönderir.

Cloud Function, Pub/Sub'dan gelen veriyi alır, doğrular ve önceki veriye göre %5'ten az değişiklik varsa kaydetmez. Geçerli veriler Firestore ve BigQuery'ye kaydedilir, hatalar ise loglanarak yönetilir. Bu filtreleme yöntemi, gereksiz veri girişini engelleyip depolama maliyetini azaltır.

### 3.3 Veri Depolama Stratejisi

#### Firestore Koleksiyonu (sensor\_readings)

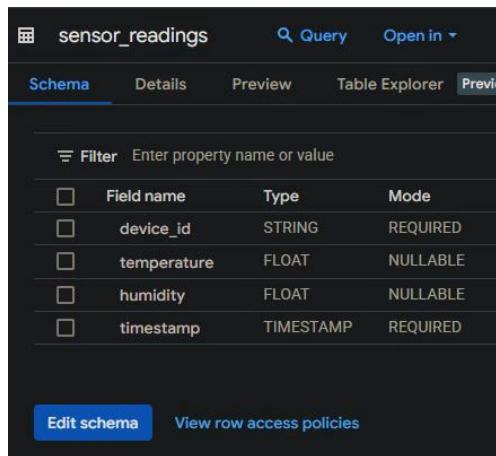


A screenshot of the Google Cloud Platform Firestore interface. It shows a single document with the ID '00qw9iCO1kFvdPRUPDWD'. The document contains the following data:

```
// Döküman yapısı:
{
 temperature: Number,
 humidity: Number,
 device_id: String,
 timestamp: DateTime
}
```

The document also displays its properties: device\_id: "sensor-001", humidity: 44.6, temperature: 28.92, and timestamp: May 29, 2025 at 7:50:05.000 AM UTC+3.

#### BigQuery Tablosu (sensora\_dataset.sensor\_readings)



A screenshot of the Google Cloud Platform BigQuery schema editor. It shows the schema for the 'sensora\_dataset.sensor\_readings' table:

| Field name  | Type      | Mode     |
|-------------|-----------|----------|
| device_id   | STRING    | REQUIRED |
| temperature | FLOAT     | NULLABLE |
| humidity    | FLOAT     | NULLABLE |
| timestamp   | TIMESTAMP | REQUIRED |

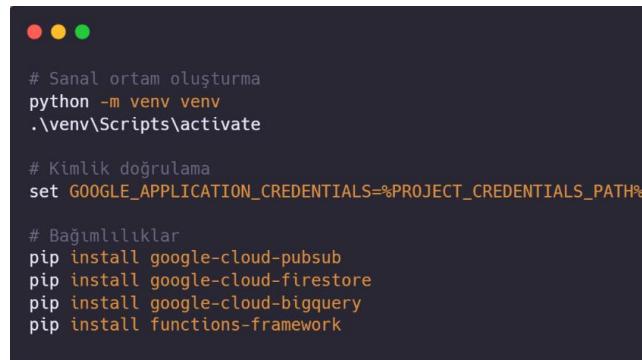
At the bottom, there are buttons for 'Edit schema' and 'View row access policies'.

Sensör verileri, hızlı erişim için Firestore koleksiyonunda saklanır. Aynı veriler, detaylı analiz için BigQuery'de tablo olarak tutulur. Böylece gerçek zamanlı işlem ve uzun dönemli analiz dengelenir.

---

## 4. DEPLOYMENT VE TEST SÜREÇLERİ

### 4.1 Geliştirme Ortamı Kurulumu



```
Sanal ortam oluşturma
python -m venv venv
.\venv\Scripts\activate

Kimlik doğrulama
set GOOGLE_APPLICATION_CREDENTIALS=%PROJECT_CREDENTIALS_PATH%

Bağımlılıklar
pip install google-cloud-pubsub
pip install google-cloud-firestore
pip install google-cloud-bigquery
pip install functions-framework
```

Geliştirme ortamı için Python sanal ortamı oluşturulur ve aktif hale getirilir. Gerekli Google Cloud kütüphaneleri (pubsub, firestore, bigquery) ile functions-framework paketi yüklenir.

Ayrıca, kimlik doğrulama için servis hesabı anahtarı ortam değişkeni olarak ayarlanır. Bu adımlar, uygulamanın bulut servislerine güvenli ve sorunsuz bağlanması sağlar.

## 4.2 Cloud Function Deployment

```
Deployment komutu ve çıktı:
%PATH%\sensora_app\cloud_function_code>
gcloud functions deploy process_sensor_data
--entry-point process_sensor_data
--runtime python313
--trigger-topic iot-sensor-data
--region europe-west3
--project sensora-460619

Başarılı deployment:
✓ [Build] Build ID: 6875294f-8e7d-4958-9e39-ffab4954bf53
✓ [Service] Cloud Run service oluşturuldu
✓ [Trigger] Pub/Sub trigger yapılandırıldı
```

## Deployment Özellikleri:

- Runtime: Python 3.13
- Memory: 256MB
- Timeout: 60 saniye
- Max Instances: 6
- Generation: 2nd Gen (Varsayılan)

Cloud Function, Python 3.13 runtime ile belirtilen bölgede ve iot-sensor-data Pub/Sub konusu tetikleyicisiyle dağıtılmıştır.

Başarılı deploy sonrası Cloud Run servisi oluşturulur ve trigger aktif hale getirilir. Fonksiyon, 256MB bellek, 60 saniye zaman aşımı ve maksimum 6 instance ile çalışır.

Bu yapılandırma, güvenilir ve ölçülebilir veri işleme sağlar.

## 4.3 Test ve Doğrulama

### Fonksiyon Testleri

1. **Local Test:** Functions Framework ile yerel test
2. **Integration Test:** Pub/Sub mesaj gönderimi
3. **End-to-End Test:** Veri akışının tamamı

### Log Analizi

```
Cloud Function logları
gcloud functions logs read process_sensor_data --region=europe-west3

Örnek başarılı log:
"Received message: {'temperature': 25.34, 'humidity': 52.18, 'device_id': 'sensor-001'}"
"Data saved to Firestore. Document ID: abc123..."
"Data inserted into BigQuery successfully."
```

Fonksiyonun doğruluğu üç aşamada test edilir. Öncelikle, Functions Framework kullanılarak yerelde çalışması kontrol edilir. Ardından, Pub/Sub'a mesaj gönderilerek entegrasyon testi yapılır.

Son olarak, veri akışının tamamı test edilerek uçtan uca doğrulama sağlanır.

Testler sırasında oluşan loglar, gcloud functions logs read komutuyla incelenir. Başarılı örnek loglar, mesaj alındığını ve verinin Firestore ile BigQuery'ye başarılı şekilde kaydedildiğini gösterir.

---

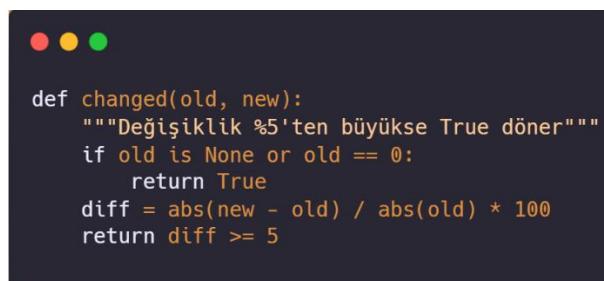
## 5. PERFORMANS VE OPTİMİZASYON

### 5.1 Sistem Performansı

Test ortamında mesaj işleme süresi yaklaşık 200-500 ms arasında değişir ve saniyede 5-10 mesaj işlenebilir. Hata oranı testlerde %1 olarak gözlemlenmiş olup, Cloud Functions SLA kapsamında %99.9 üzeri erişilebilirlik sağlanır.

### 5.2 Optimizasyon Stratejileri

#### Akıllı Veri Filtreleme



```
def changed(old, new):
 """Değişiklik %5'ten büyükse True döner"""
 if old is None or old == 0:
 return True
 diff = abs(new - old) / abs(old) * 100
 return diff >= 5
```

#### Hata Yönetimi

- Try-catch blokları ile exception handling
- Detaylı loglama ve monitoring
- Pub/Sub retry mekanizması (RETRY\_POLICY\_DO\_NOT\_RETRY)

Sistemde akıllı veri filtreleme kullanılarak, önceki veriye göre %5'ten az değişiklik gösteren veriler kaydedilmez.

Hata yönetimi için try-catch bloklarıyla istisnalar yakalanır, detaylı loglama yapılır ve Pub/Sub mesajlarının tekrar denemeleri kontrol edilir.

Bu sayede performans artarken hata yönetimi ve izleme güvence altına alınır.

---

## 6. GÜVENLİK VE YETKİ YÖNETİMİ

Proje, minimal yetki prensibine uygun olarak IAM rolleri ve servis hesaplarıyla yönetilir;

Cloud Functions, Pub/Sub, Firestore ve BigQuery erişimleri ayrı ayrı yetkilendirilir ve kimlik doğrulama JSON anahtar dosyasıyla sağlanır.

### 6.1 IAM Rollerı

- **Cloud Functions Invoker:** Pub/Sub trigger için
- **Pub/Sub Publisher:** Veri gönderimi için
- **Firestore User:** Veritabanı erişimi
- **BigQuery Data Editor:** Veri ambarı erişimi

### 6.2 Servis Hesabı

- Compute Engine default service account kullanımı
- Minimal yetki prensibi (principle of least privilege)
- JSON key file ile kimlik doğrulama

## 7. MONİTORİNG VE LOGLAMA

### 7.1 Cloud Function Monitoring

- Invocation Count:** Fonksiyon çağrı sayısı
- Execution Time:** Ortalama çalışma süresi
- Error Rate:** Hata oranı izleme

### 7.2 Loglama Stratejisi

```
Detaylı loglama örnekleri:
print(f"Received message: {sensor_data}")
print(f"Data saved to Firestore. Document ID: {doc_ref.id}")
print("Data inserted into BigQuery successfully.")
print("Değişiklik %5 altında, veri yazılmadı.")
```

## 8. VERİ GÖRSELLEŞTİRME VE DASHBOARD

### 8.1 Looker Studio Entegrasyonu

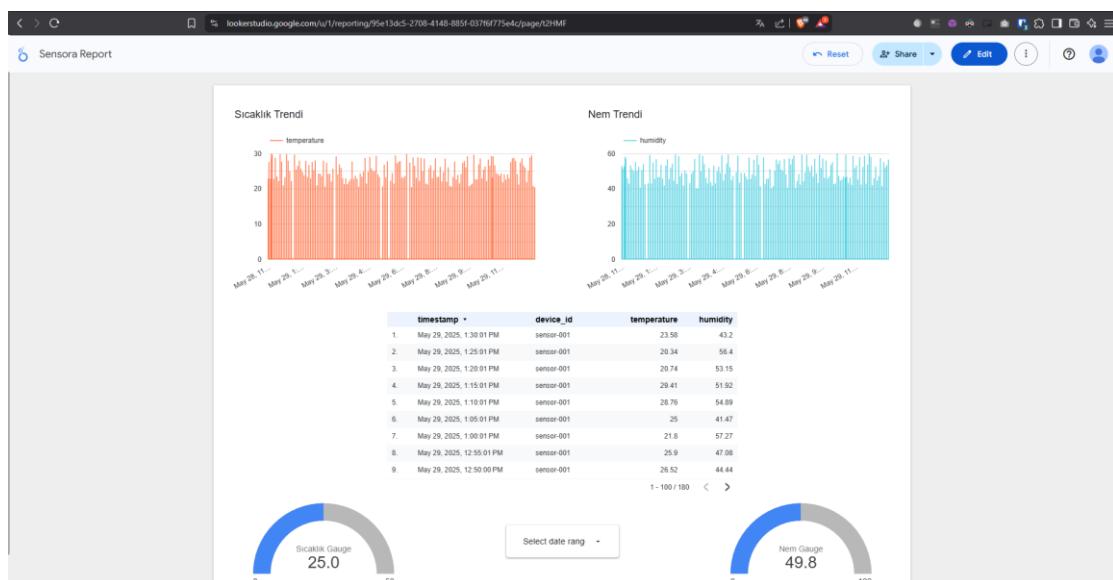
Proje kapsamında toplanan sensör verilerinin gerçek zamanlı izlenmesi ve analiz edilmesi için Google Looker Studio ile entegre bir dashboard oluşturulmuştur.

#### Dashboard Özellikleri

- Gerçek Zamanlı Veri:** BigQuery veri kaynağından canlı veri akışı
- Zaman Serisi Grafikleri:** Sıcaklık ve nem değerlerinin zaman içindeki değişimini
- Cihaz Bazlı Filtreleme:** Device ID'ye göre verifiltreleme imkanı
- Istatistiksel Özetter:** Ortalama, minimum, maksimum değerler
- Trend Analizi:** Günlük, saatlik trend görüntümeli

#### Görselleştirme Faydaları

- Operasyonel İzleme:** Sensör durumlarının anlık takibi
- Trend Analizi:** Uzun dönemli veri eğilimlerinin belirlenmesi
- Anomali Tespiti:** Normal dışı değerlerin görsel tespiti
- Raporlama:** Periyodik raporların otomatik oluşturulması



---

## 9. PROJE DOSYA YAPISI

```
sensora_app/
├── {account_key}.json # GCP Service Account Key
├── sensor_data_publisher.py # Ana veri gönderici script
├── venv/ # Python sanal ortam
├── schema.json # BigQuery tablo şeması tanımı
├── cloud_function_code/
│ ├── main.py # Ana fonksiyon kodu
│ └── requirements.txt # Python bağımlılıkları
└── publisher_function/
 ├── main.py # Alternatif publisher
 └── requirements.txt # HTTP trigger fonksiyonu
 # Bağımlılıklar
```

---

## 10. ÖĞRENİLEN DERSLER VE ÇÖZÜLEN SORUNLAR

### 10.1 Teknik Zorluklar

1. **Cloud Functions Gen2 Geçişi:** Otomatik olarak 2nd gen'e geçiş
2. **IAM Yetkilendirme:** Servis hesabı rollerinin doğru yapılandırılması
3. **Timestamp Conversion:** Unix timestamp → DateTime dönüşümü
4. **BigQuery Schema:** Tablo şemasının doğru tanımlanması

### 10.2 Çözüm Yaklaşımları

1. **Dokümantasyon Takibi:** GCP dökümanlarından aktif yararlanma
2. **Yapay Zeka Destekli Yardım:** Dokümantasyon yorumu, hata çözümü ve yapılandırma
3. **Log-Driven Development:** Detaylı loglama ile hata ayıklama
4. **Incremental Testing:** Adım adım test ve doğrulama
5. **Error Handling:** Kapsamlı exception yönetimi

---

## 11. SONUÇ VE GELECEK PLANLAR

### 11.1 Başarılı Hedefler

- ✓ Google Cloud üzerinde tam fonksiyonel IoT pipeline
- ✓ Gerçek zamanlı veri işleme ve depolama
- ✓ Kapsamlı hata yönetimi ve loglama
- ✓ Ölçeklenebilir serverless mimari
- ✓ Çoklu veri deposu entegrasyonu (Firestore + BigQuery)

## **11.2 Sistem Avantajları**

- **Serverless:** Otomatik ölçekleme, yönetim gerekmez
- **Cost-Effective:** Kullanım bazlı ücretlendirme
- **High Availability:** Cloud provider SLA garantileri
- **Real-time:** Düşük latency ile veri işleme
- **Scalable:** Yüksek throughput desteği

## **11.3 Gelecek Geliştirmeler (Roadmap)**

- **Cloud Scheduler:** Otomatik veri gönderimi
  - **Anomaly Detection:** ML tabanlı anormallilik tespiti
  - **Multi-Device Support:** Çoklu sensör desteği
  - **Real-time Alerts:** Threshold bazlı uyarı sistemi
- 

## **12. REFERANSLAR VE KAYNAKLAR**

### **12.1 Kullanılan Teknolojiler**

- **Google Cloud Functions:** Sunucusuz (serverless) hesaplama hizmeti
- **Google Cloud Pub/Sub:** Mesajlaşma servisi
- **Google Cloud Firestore:** NoSQL veritabanı
- **Google Cloud BigQuery:** Veri ambarı (data warehouse)
- **Google Cloud IAM:** Yetkilendirme ve erişim kontrolü
- **Cloud Logging (Stackdriver):** Loglama ve izleme hizmeti
- **Docker:** Uygulamaların konteyner içinde taşınabilir şekilde çalışması için
- **Python 3.13:** Backend geliştirme dili

### **12.2 Önemli Linkler**

- [Cloud Run Functions - Resmi Dökümantasyon](#)
  - [Pub/Sub - Resmi Dökümantasyon](#)
  - [Firestore - Resmi Dökümantasyon](#)
  - [BigQuery - Resmi Dökümantasyon](#)
  - [IoT ve Pub/Sub Entegrasyonu - ChirpStack](#)
  - [Firestore ve BigQuery Entegrasyonu - GCP Rehberi](#)
-