

KARADENİZ TEKNİK ÜNİVERSİTESİ
OF TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ



VERİ TABANI YÖNETİM SİSTEMLERİ
PROJE RAPORU

İLAYDA KARBUZ (414452)

2023 -2024 BAHAR DÖNEMİ

İÇİNDEKİLER

- 1. Proje Konusu ve Amacı**
 - 1.1. Proje Konusu
 - 1.2. Proje Amacı
- 2. Proje İçeriği**
- 3. Couchbase Yapısı**
- 4. Couchbase ile Android uygulama geliştirirken kullanılan hizmetler**
 - 4.1. Couchbase Server
 - 4.1.1. Bağlantı kurulumu
 - 4.1.2. Temel mantığı ve UI işlemler
 - 4.2. Couchbase Sync Gateway
 - 4.2.1. Bağlantı kurulumu
 - 4.2.2. Temel mantığı
 - 4.3. Couchbase Lite
- 5. Proje İlerleyişi**

1. Proje Konusu ve Amacı

- 1.1. Proje Konusu: Couchbase ile Android Uygulama Geliştirme
- 1.2. Proje Amacı: Couchbase kullanarak basit şekilde kullanıcı oluşturan, kullanıcının kitap eklemesi yapmasına imkan tanıyan ve eklediği kitapları görüntüleyebildiği temel düzey bir Android uygulaması geliştirmek.

2. Proje İçeriği

Bu projede,

- Couchbase NoSQL veri tabanı yönetim sisteminin Android için sağlamış olduğu hizmetler olan Couchbase Lite, Couchbase Sync Gateway kullanılmış ayriyeten veriler bulut hizmeti olan Couchbase Server’da depolanmıştır.
- Uygulamada üyelik söz konusudur. Bu yüzden kullanıcı oluşturulup giriş yapılması gerekir.
- Kullanıcı oluşturma sonrası veritabanında hali hazırda bulunan kitapların sıralandığı bir main sayfası kullanıcıyı karşılar.
- Bu main sayfasında ayriyeten kullanıcının kitap eklemesine yarayan sayfaya yönlendirme amaçlı bir buton da bulunmaktadır.
- Kitap ekleme sayfasında kullanıcı istediği alanı doldurup istemediğini doldurmamakta serbesttir. Veriler doldurulan alana göre çekilir ve veri tabanına sadece doldurulan alanlara dair bilgiler kaydedilir.
- Main sayfasında gösterilen kitap bilgilerinde sadece eklenen bilgiler gösterilir. Esnek yapılıdır. Her bilgi kendine ait hücrede gösterilir.

3. Couchbase Yapısı

Couchbase NoSQL bir veritabanıdır. Cap teoremindeki duruşu kullanım şekline bağlı olarak çeşitlenmektedir. Sadece bir clusterın kullanıldığı durumlarda CP’ye denk düşerken XDCR ile beraber birden çok clusterın aynı veritabanına yazım ve veritabanından okuma yapması durumunda ise AP’ye denk düşer. Buradan da anlaşılabilirdiği üzere partition tolerance bu veritabanı yapısının vazgeçilmezidir.

NoSQL veritabanı olmasının avantajlarının yanında community edition versiyonunda çoğu özelliğin kısıtlanması ve özellikle de dökümantasyon eksikliği ile örnek bulunmasının zorluğu bu veritabanının öne çıkmasını engelleyen dezavantajlarıdır.

4. Couchbase ile Android uygulama geliştirirken kullanılan hizmetler

4.1. Couchbase Server

4.1.1. Bağlantı kurulumu

Serverın kurulumu sonrası tercih edilen herhangi bir arama motorunda “localhost:8091” yazılarak cluster oluşturma ve giriş ekranına erişilmektedir. Yani server 8091 portunda çalışmaktadır. Bu yüzden bu portun boş ve erişilebilir olması gerekmektedir.

İlk kurulum sonrası karşımıza çıkan ekranda ya yeni bir cluster oluşturulması ya da varolan bir cluster'a girilmesi istenir.



İlgili bu ekranda yeni cluster oluşturulduğunda

Couchbase > New Cluster

Cluster Name

Create Admin Username

Administrator

Create Password

6 characters minimum

Confirm Password

6 characters minimum

< Back

Next: Accept Terms

çıkan bu ekranda cluster ismi, cluster admin rolü için bir kullanıcı adı ve şifreler verilmelidir.

Couchbase > New Cluster > Configure

Host Name / IP Address

127.0.0.1

enable node-to-node encryption

IP Family Preference

IPv4 IPv6 IPv4-only IPv6-only

Service Memory Quotas

Data 256 MIB

Query

Index 512 MIB

Search 256 MIB

Analytics 1024 MIB

Eventing 256 MIB

Backup

TOTAL QUOTA 2304MIB

RAM Available 985MIB Max Allowed Quota 788MIB

Index Storage Setting

Standard Global Secondary

Memory-Optimized

Data Disk Path

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Indexes Disk Path

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Eventing Disk Path

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Analytics Disk Paths

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Java Runtime Path

optional

< Back

Save & Finish

İlerleyen adımda serverın özellikleri ve sahip olması istenenler belirtilir. Eğer yerel geliştirme yapılacaksa host name kısmı 127.0.0.1 olarak bırakılabilir. Ancak dağıtık şekilde birçok node eklenecekse host name yerine ilgili makinenin IP adresinin konulması ilerde düğüm eklenmesi ve verilere her yerden ulaşılabilmesi için daha sağlıklıdır. Biz yerel bir geliştirme yapacağımızdan ve bazen direkt IP adresi verildiğinde uygulamada bağlantı hataları oluşturabildiğinden aynen bıraktık.

Couchbase Server'da ilerleyen süreçte kullanacağımız Sync Gateway'e bağlantı sağlayabilmek için ilk adımda Security kısmından bir kullanıcı oluşturmak gerekir. Bunu Direkt Web Console veya REST API ile yapabilmekteyiz. Biz Web Console üzerinden oluşturduk.

Add New User

Username

ilayda

Full Name (optional)

Password

Verify Password

Administrative

Full Admin

Read-Only Admin

Bucket

Application Access

bucket...

Cancel

Add User

Full Admin: Cluster üzerinde oluşturulan tüm bucketlara erişip yazma, okuma yapabilen ve web console'a erişebilen kullanıcılardır.

Read-Only Admin: Cluster üzerinde bucketlara okuma için erişebilen ancak yazamayan, onun dışında web console'a ulaşp istatistikleri görebilen kullanıcılarıdır.

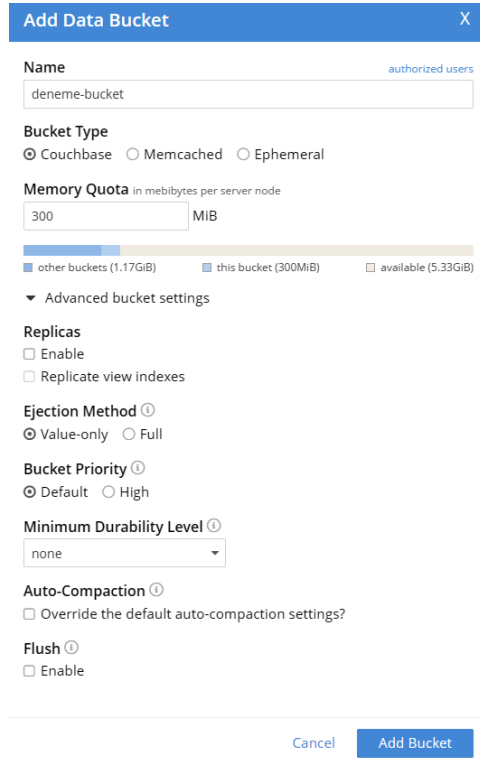
Bunlara ek olarak Bucket altında Application Access seçeneği ile kullanıcıların erişebilecekleri bucketlar da seçilebilir. Web Console'a erişmesinin istenmediği uygulama bazlı kullanıcılar için kullanılabilir yerdir.

4.1.2. Temel Mantığı ve UI İşlemler

Couchbase Server'ın verileri depolama şekline bakacak olursak,

- **BUCKET:**

İlişkisel vtyslerinde veritabanlarına karşılık düşen yapılarıdır.

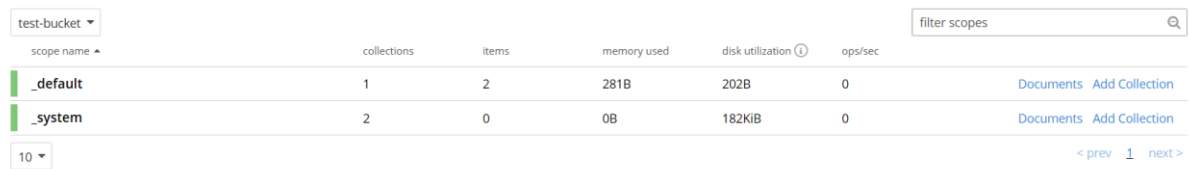


Bucketı isimlendirdikten sonra bu bucketa ne kadar depolama alanı vermek istediğimizi belirtebiliyoruz.

Advanced bucket settings kısmında, Replicas: Özellikle birden fazla nodeluk sistemlerde kullanılan bu özellik bucketa eklenen verilerin ne kadar sayıda çoğaltılacağını seçebilmemizi sağlar. Bizim sistemimizde tek node üzerinde ilerlediğimiz için replicas enable seçeneğini seçmiyoruz.

- **SCOPES:**

İlişkisel vtyslerinde schema dediğimiz yapıya denk düşer.



scope name	collections	items	memory used	disk utilization	ops/sec	
_default	1	2	281B	202B	0	Documents Add Collection
_system	2	0	0B	182KiB	0	Documents Add Collection

Bir bucket oluşturulduğu anda yukarıda gösterildiği gibi iki ayrı scope direkt oluşturulur.

_default scope: Herhangi bir yetkilendirme gerekmeyen işlemlere dair dökümanları içerisinde barındıran ve her kullanıcının erişmesi beklenen scopetur.

_system scope: Bu scope'a adminler dahil kimse erişmemelidir. Sistemin işleyişi ile ilgili bilgileri içerisinde tutar.

Bunların dışında bir özel scope da kurulabilir. Scope'u doğru bucketta oluşturmaya dikkat edilmelidir. İstenen kullanıcılara erişim verilebilir.

- **COLLECTION:**

İlişkisel vtyslerinde tablolara denk düşer.

İçerisine collection eklenmesi istenen scope belirtilmelidir.

- **DOCUMENT:**

İlişkisel vtyslerde satırlara denk düşer. Couchbase dökümanları JSON formatında depolar.

4.2. Couchbase Sync Gateway

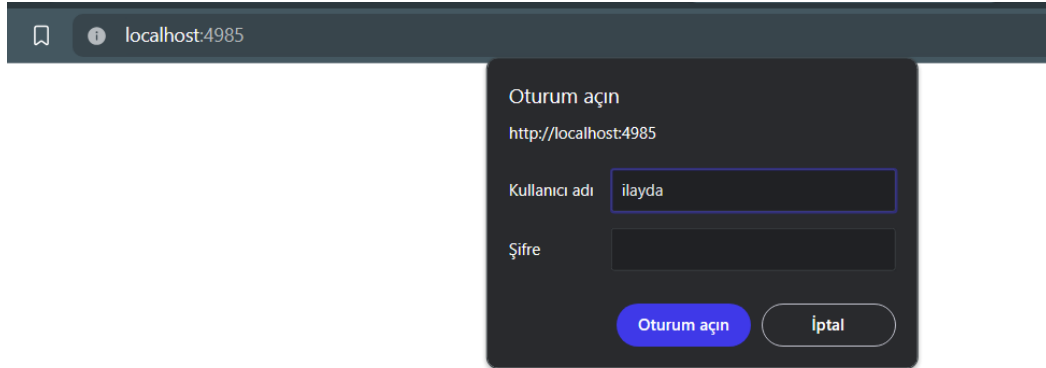
Couchbase uygulamalarında direkt server üzerinden işlem yapılması önerilmemektedir. Bu yüzden hem erişimleri biraz daha kontrollü sağlamak hem de özellikle Android uygulamalarda verilerin sadece cihazda tutulmaması ve farklı cihazlardan da eş zamanlı şekilde verilere ulaşıp güncellemesini için kullanılır. Bir yandan senkronize edildiği için en güncel veriler eş zamanlı olarak kullanıcının önüne düşebilir.

4.2.1. Bağlantı Kurulumu

Sync Gateway'in indirilmesi sonrası web consoleuna erişebilmek için "localhost:4985 | 4984 | 4986" tarayıcıya yazılabilir. Localhost sonrasına eklenen port numaralarının erişilebilir olmasına dikkat edilmelidir.

4984: public sayfa içindir, Erişim kontrol edilmez.

4985: admin sayfasıdır. Sync Gateway konfigürasyon dosyasında belirtilen admin bilgileriyle giriş yapılabilen sayfadır.



4986: metrik sayfasıdır. Gateway üzerinde yapılan iş akışları takip edilebilir.

Sync Gateway'in kullanım amacı cloud servera verileri gönderebilmek ve oradan veri getirebilmek olduğundan servera bağlanması gerekir. Bunu sağlamak için konfigürasyon dosyasının düzenlenmesi gerekir. Bu işlem aynı zamanda REST API tarafından da yapılabilir. Kendi oluşturduğum örnek konfigürasyon dosyası üzerinden anlatmamız gerekirse,

```
{
  "disable_persistent_config": true,
  "adminInterface": "127.0.0.1:4985",
  "interface": "0.0.0.0:4984",
  "server_tls_skip_verify": false,
  "use_tls_server": false,
  "databases": {}
  "db": {
    "server": "walrus:data",
    "users": {
      "GUEST": {"disabled": false, "admin_channels": ["*"]}
    },
    "allow_conflicts": false,
    "revs_limit": 20,
    "num_index_replicas": 0
  },
  "notesdb": {
    "bucket": "notes-bucket",
    "server": "http://127.0.0.1:8091",
    "username": "ilayda",
    "password": "password",
    "users": {
      "GUEST": {"disabled": false, "admin_channels": ["*"]}
    },
    "allow_conflicts": false,
    "revs_limit": 20,
    "num_index_replicas": 0,
    "import_docs": true
  },
}
```

```
"testdb": {
  "bucket": "test-bucket",
  "server": "http://127.0.0.1:8091",
  "username": "ilayda",
  "password": "password",
  "users": {
    "GUEST": {"disabled": false, "admin_channels": ["*"]}
  },
  "allow_conflicts": false,
  "revs_limit": 20,
  "num_index_replicas": 0,
  "import_docs": true
},
"booksdb": {
  "bucket": "books-bucket",
  "server": "http://127.0.0.1:8091",
  "username": "ilayda",
  "password": "password",
  "users": {
    "GUEST": {"disabled": true},
    "ilayda": {"password": "password", "admin_channels": ["*"]}
  },
  "allow_conflicts": false,
  "revs_limit": 20,
  "num_index_replicas": 0,
  "import_docs": true,
  "enable_shared_bucket_access": true
},
"allow_conflicts": false,
"revs_limit": 20,
"num_index_replicas": 0,
"import_docs": true,
"sync": "function(doc, oldDoc) { if (doc.type == 'users') { channel('users'); }}"
}
```

server_tls_skip_verify: güvenli server kullanımı ile ilgilidir. Biz http üzerinden ilerlediğimiz için bu değeri false olarak setliyoruz.

Use_tls_server: yukarıdaki durumla aynıdır.

databases keyi ile beraber sync gatewayde kurmak istediğimiz veri tabanları ve bunları bağlamak istediğimiz serverlar, bucketlar belirtilmeye başlanır.

ÖNEMLİ BİLGİ: Veritabanı oluşturma sebebimiz, sync gatewayin servera bağlanımının kendisinde oluşturulan veritabanının kendisine söylenen server bucketına denk düşmesi gerekmesindendir. Yani sync gateway veritabanı server bucketına bağlanır.

Örnek içerisindeki “db” server keyinde belirtilen “walrus:data” ile belirtildiği üzere çalıştırıldığı makinenin belleği üzerinde geçici olarak oluşturulmuş, herhangi bir gerçek serverla bağlantısı olmayan bir veritabanıdır.

İlerleyen satırlardaki veritabanları ise,

server keyine denk düşen 127.0.0.1:8091 clusterına bağlanır. Bu cluster bizim couchbase serverımızı açan, oraya bağlanan linktir. Özelleştirilmiş IP bazlı olanlarda da IP değerinin 127.0.0.1 yerine yazılması gerekir.

bucket keyine denk düşen değere de veritabanının temsil edeceği ve bağlanacağı serverdaki ilgili bucket ismi verilir.

username ile serverı kurduktan sonra oluşturduğumuz full admin haklarına sahip kullanıcının ismi belirtilir.

password ile bu full admin kullanıcının şifresi verilir.

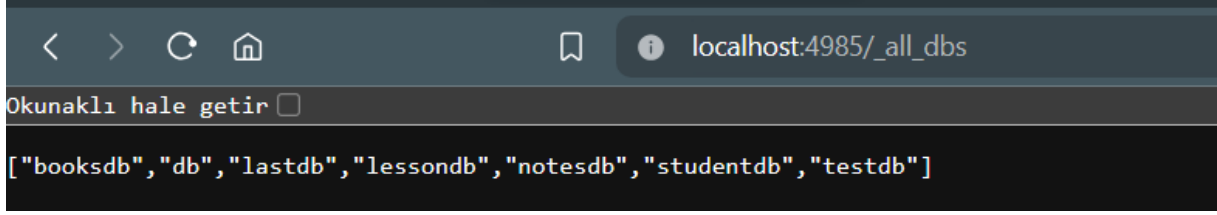
users ile bu veritabanına bağlanabilecek çeşitli kullanıcılar ve onların yetki düzeyleri belirtilir.

num_index_replicas ile veritabanının kaç kopya ile bağlanacağı belirtilir. Bizim tek nodeumuz olduğundan 0 olarak atamamız gerekir. Yoksa gerekli indexler oluşturulmadığından bağlantı hataları alınır.

import_docs ilgili veritabanının kullanıcılara erişim için açılıp açılmadığını belirtir. Community Edition’da default değeri false olduğundan veritabanlarına erişim problemi yaşamamak için bu keyin açılıp değerinin true verilmesi gerekir.

Gerekli değişiklikler sonrası konfigürasyon dosyası kaydedilir ve Sync Gateway yeniden başlatılır. 1

localhost:4985/_all_dbs şeklinde tarayıcıya yazılıp ilgili link açıldığında konfigürasyonda oluşturduğumuz veritabanlarının gözükmesi gerekir.



Aynı zamanda servera başarılı şekilde bağlanıldığını cloud server web consoleunda (localhost:8091) ilgili bucketın index değerlerine ve aktif öge sayılarına bakarak anlayabiliriz. Bucket bağlantı sağlanamadığı ve yeni oluşturulduğunda aktif öge sayısı sıfırdır ve indexleri bulunmamaktadır.

Sadece guest user oluşturulması durumunda 5, ekstra kullanıcı tanımlanması durumunda 10 index oluşmuş olması gerekir.

sadece guest user oluşturulmuş

Bucket & Scope							
test-bucket		_default		view by index		filter indexes...	
index name	requests/sec	resident ratio	items	data size	keyspace	status	
sg_access_1	0	0%	0	104KiB	test-bucket_default_default	ready	
sg_allDocs_1	0	0%	0	104KiB	test-bucket_default_default	ready	
sg_channels_1	0	0%	0	104KiB	test-bucket_default_default	ready	
sg_roleAccess_1	0	0%	0	104KiB	test-bucket_default_default	ready	
sg_syncDocs_1	0	0%	2	108KiB	test-bucket_default_default	ready	

Ekstra kullanıcı oluşturulmuş

Bucket & Scope							
books-bucket		_default		view by index		filter indexes...	
index name	requests/sec	resident ratio	items	data size	keyspace	status	
sg_access_1	0	0%	0	146KiB	books-bucket_default_default	ready	
sg_access_x1	0	0%	0	146KiB	books-bucket_default_default	ready	
sg_allDocs_1	0	0%	0	147KiB	books-bucket_default_default	ready	
sg_allDocs_x1	0	0%	5	149KiB	books-bucket_default_default	ready	
sg_channels_1	0	0%	0	146KiB	books-bucket_default_default	ready	
sg_channels_x1	0	0%	0	146KiB	books-bucket_default_default	ready	
sg_roleAccess_1	0	0%	0	146KiB	books-bucket_default_default	ready	
sg_roleAccess_x1	0	0%	0	146KiB	books-bucket_default_default	ready	
sg_syncDocs_1	0	0%	54	178KiB	books-bucket_default_default	ready	
sg_syncDocs_x1	0	0%	54	178KiB	books-bucket_default_default	ready	
sg_tombstones_x1	0	0%	0	147KiB	books-bucket_default_default	ready	

Başarılı bağlantı sonrası artık Sync Gateway üzerinden cloud server işlemlerini gerçekleştirebilir hale geliriz.

4.2.2. Temel Mantığı

Sync Gatewayde işlemler url endpointleri ile ilerlemektedir. Önceki adımda belirttiğim _all_dbs örneğinde olduğu gibi endpointler ile çeşitli işlemler gerçekleştirebiliyoruz. Bu işlemler aynı zamanda REST API üzerinden herhangi bir http client aracılığıyla da yapabilmekteyiz. Bu alanda özellikle Couchbase Server dökümantasyonunda Admin RestAPI sayfasında kullanılabilecek tüm endpointler ve

bunları rest api ile kullanırken nasıl bir yapı oluşturulması gerektiği anlatılmıştır. Aynı zamanda yapılmak istenen işleme ait yerde istenen yerler doldurulduğunda otomatik bir curl request komutu oluşturulmaktadır. Bunlardan da yararlanarak command promptta işlemler gerçekleştirilebilir.

Sync Gateway’de işlemler erişimler bu yüzden de kullanıcılar üzerinden ilerlemektedir. Bu nedenle user ve channel yapıları oluşturulmuştur.

User tanımlanarak ilgili veritabanı üzerinde hangi kullanıcının işlem yapabileceği belirtilir.

Channel tanımlanarak ise ilgili veritabanında işlem yapabilen bu kullanıcıların hangi erişim seviyesine sahip olacağı belirtilebilir. Örneğin bir okul veritabanında öğrenci ve öğretmen diye iki farklı channel oluşturulup bu iki channela birbirinden farklı yetkiler verilebilir. Böylelikle istenmeyen durumlardan kaçınılabilir.

Konfigürasyon dosyasında verilen *sync* keyinde bu channelara ait erişim düzeylerine ilişkin dosya işlemleri belirtilir.

4.3. Couchbase Lite

Mobil işlemlerini gerçekleştirmek için bir kütüphanedir diyebiliriz. Yapacağımız işlemleri bu kütüphane içindeki metotlarla gerçekleştiriyoruz.

Buradan sonraki yerleri projeye devam ettirmenin daha sağlıklı olduğunu düşündüğümüzden projeye ait kodlar ve uygulamalarla devam ediyoruz.

5. PROJE İLERLEYİŞİ

Kullanıcı oluşturulmak istendiğinden bir SignUp sayfası oluşturulmuştur.

Http clientlarla işlem yapacağımız için bolca request oluşturacağız.

```
usage
public void kullanıcıOlustur(String kullaniciad, String sifre, String email) {
    OkHttpClient httpClient = new OkHttpClient();
    String uid = UUID.randomUUID().toString();
    String json = "{\"name\":\"" + kullaniciad + "\",\"password\":\"" + sifre + "\",\"admin_channels\":[\"users\"],\"" +
        "\"email\":\"" + email + "\",\"uid\":\"" + kullaniciad + "\"}";
    RequestBody body = RequestBody.create(MediaType.parse("application/json"), json);

    //sync gateway üzerinde kullanıcı oluşturur ki replicator oluşturulsun ve servera erişebilsin
    Request request = new Request.Builder()
        .url("http://10.0.2.2:4985/booksdb/_user/")
        .post(body)
        .build();

    httpClient.newCall(request).enqueue(new Callback() {...});
}
```

Sync Gateway’de hali hazırda user endpointi olduğundan onun üzerinden kullanıcıyı ekleyebiliyoruz. Sonrasında bu bilgilerle Basic Authentication yöntemiyle kullanıcı oturumu başlatabileceğiz.

localhost:4985/booksdb/_user/

Okunaklı hale getir

```
[{"08fd7487-6003-463d-a7a2-49815c175f57","34f33ac0-aae3-47d5-b6b4-ace94e43c20a","3ddb5044-60e4-4909-96ef-94ffd10b87b3","44ead31b-2a26-43ff-ab04-6a5e5d0b871f","51f6edd4-6e17-4aaf-98be-2c3c275d843c","547aead8-8013-4c6c-a61f-901b56b8473b","6115c712-0c6f-41c4-ab72-e5a0ade66595","63b65f75-81cb-4bf9-8243-ebd7ac078c19","793a6717-a01e-4cb4-90ca-54044563e85f","883f2e92-fd9c-40f1-a3cf-3d575b048a3d","8b1b0bc6-24db-4ae3-bcf0-d3abacafdf3c","9acdb599-bb6e-4af7-93bc-80a22ae8fcb8","a0316f58-257e-4859-8c73-1c25d79acf8b","a1db4dd9-db29-48e0-a6fc-41f42468b98e","bdf5219-7b1d-4d04-a948-231ef698cf8","book_ilayda","books","cbdc4aea-9275-46ad-8baf-20a497a5dafc","d314c51d-4942-4529-86a5-220885f71156","e050fb27-e1fe-4e64-bf75-fd03a2e73904","ea179761-1b3b-44ff-bc4f-36044f6e7959","f13cf1ab-e5cc-4268-84a4-c6ebf0ba18a3","f61775b6-0587-40a3-a506-b261f72ad4f6","ilayda","kitap"]
```

```

1 usage
public void kullanıcıKontrol(String email, String password, Callback callback) {
    OkHttpClient client = new OkHttpClient();

    MediaType mediaType = MediaType.parse("application/json");
    //String requestBody = "{\"useremail\": \"\" + email + \"\"}";
    String requestBody = "{\"email\": \"\" + email + \"\", \"password\": \"\" + password + \"\"}";

    Request request = new Request.Builder()
        .url("http://10.0.2.2:4985/booksdb/_session")
        .post(RequestBody.create(mediaType, requestBody))
        .build();

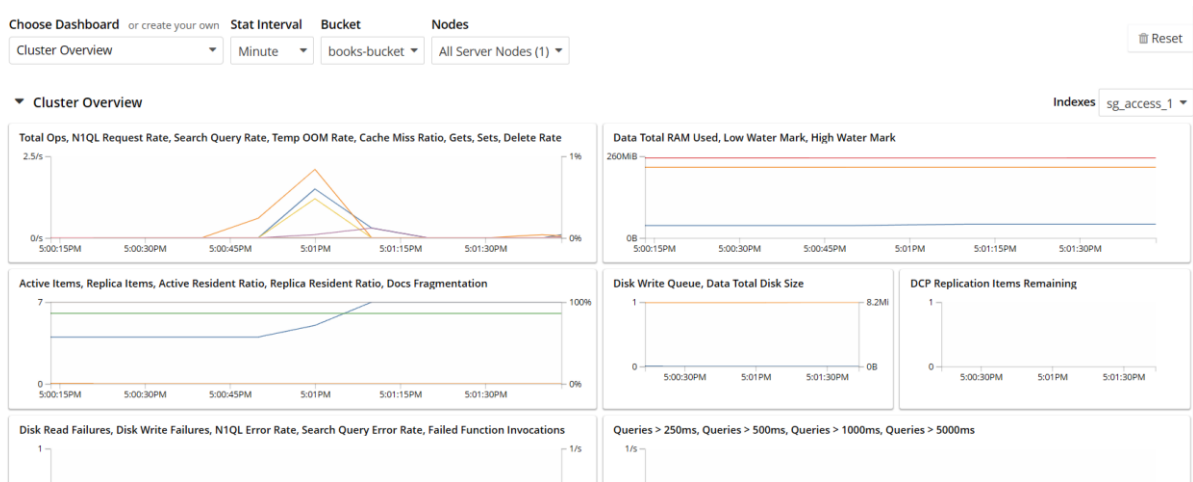
    client.newCall(request).enqueue(callback);
}

```

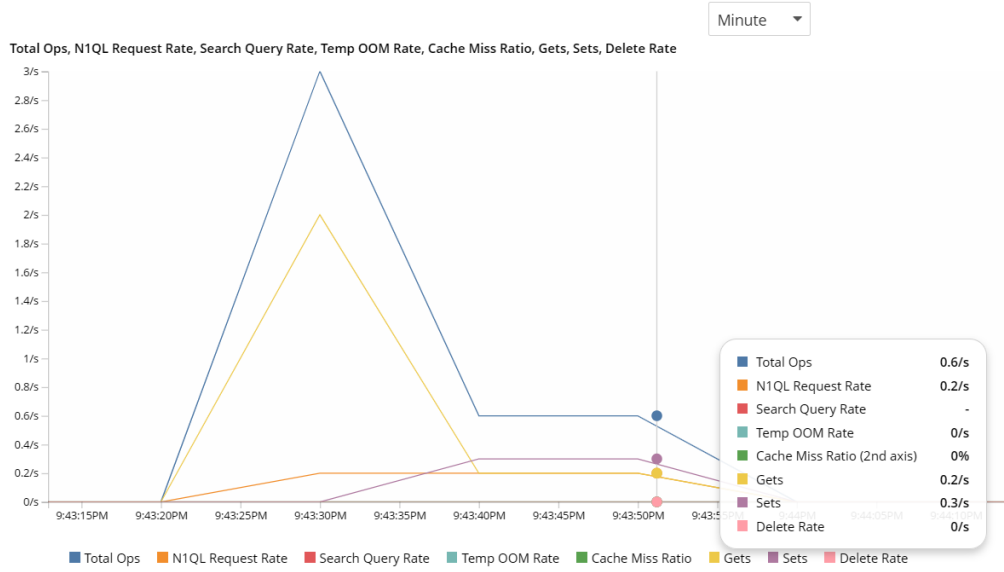
Login esnasında ise sync gatewayin bize sağladığı session endpointini kullanabiliyoruz. Burada kullanıcının maili ve şifresiyle bir string oluşturup bunu request bodye koyuyoruz. Bu esnada post request yollayarak kullanıcıyı kontrol ediyoruz.

ÖNEMLİ NOT!!!













url'deki veritabanı bağlantısı normalde 127.0.0.1 ile gösterilerek yapıyordu ancak biz bu kodları emülatörde deneyeceğimizden ve emülatör de 127.0.0.1'i aynı bilgisayar gibi kendi üzerinden server olarak aradığı için bağlantı problemi veriyor. Bu türden durumlarda emülatörde çalışırken üzerinde çalıştığı makinenin yerel servisini görmesini 10.0.2.2 şeklinde belirterek sağlıyoruz.



Yukarıdaki ekran kullanıcının başarılı bir şekilde sync gateway aracılığıyla oluşturulup clouda'a atılabildiğini gösteriyor.



Kullanıcı bilgilerinin documents halinde cloud serverda gözükmesi

   	_syncuser:ilayda	{ "name": "ilayda", "admin_channels": {"*": 2}, "all_channels": {"*": 1, "users": 2}, "sequence": 2, "passwordhash_bcrypt": "JDJhDEWJD1yNVFuNjYxNWtkc2MxbVzT21BaC53aWluMzFIUm1BL3g4aGw0Um5xN0FJQmVVG3PTm9X", "jwt_last_updated": "0001-01-01T00:00:00Z", "rolesSince": {}, "session_uuid": "a3d9350b-7856-4178-bda4-96e72c60c1e0" }
   	_syncuser:kitap	{ "name": "kitap", "admin_channels": {"users": 27}, "all_channels": {"*": 1, "users": 27}, "sequence": 27, "email": "ilayda@test.co", "passwordhash_bcrypt": "JDJhDEWJD1yNVFuNjYxNWtkc2MxbVzT21BaC53aWluMzFIUm1BL3g4aGw0Um5xN0FJQmVVG3PTm9X", "jwt_last_updated": "0001-01-01T00:00:00Z", "rolesSince": {}, "session_uuid": "a3d9350b-7856-4178-bda4-96e72c60c1e0" }
   	_syncuser:sahaf	{ "name": "sahaf", "admin_channels": {"users": 33}, "all_channels": {"*": 1, "users": 33}, "sequence": 33, "email": "sahaf@sahaf.com", "passwordhash_bcrypt": "JDJhDEWJD1yNVFuNjYxNWtkc2MxbVzT21BaC53aWluMzFIUm1BL3g4aGw0Um5xN0FJQmVVG3PTm9X", "jwt_last_updated": "0001-01-01T00:00:00Z", "rolesSince": {}, "session_uuid": "a3d9350b-7856-4178-bda4-96e72c60c1e0" }

Windows'u Etkinleştir

İlgili dökümanlardan birine girildiğinde elde içeriği göürebiliyor.

Edit Document X

_sync:user:ilayda Q

Data Metadata

```
1 {  
2   "name": "ilayda",  
3   "admin_channels": {  
4     "*": 2  
5   },  
6   "all_channels": {  
7     "*": 1,  
8     "*": 2  
9   },  
10  "sequence": 2,  
11  "passwordhash_bcrypt":  
    "JDJhDEWJD1yNVFuNjYxNWtkc2MxbVzT21BaC53aWluMzFIUm1BL3g4aGw0Um5xN0FJQmVVG3PTm9X",  
12  "jwt_last_updated": "0001-01-01T00:00:00Z",  
13  "rolesSince": {},  
14  "session_uuid": "a3d9350b-7856-4178-bda4-96e72c60c1e0"  
15 }
```

Cancel Save

Main sayfasında kitap ekleme sayfasına gönderilip kitap eklenebildiği söylenmişti. Bu esnada,

```
Kitap kitap = new Kitap();
MutableDocument doc = new MutableDocument();
doc.setString("type", "book");

if(!kad.isEmpty()){
    kitap.setBaslik(kad);
    doc.setString("baslik", kitap.getBaslik());
}
if(!yad.isEmpty()){
    kitap.setYazar(yad);
    doc.setString("yazar", kitap.getYazar());
}
if(!kat.isEmpty()){
    kitap.setKategori(kat);
    doc.setString("kategori", kitap.getKategori());
}
if(!basta.isEmpty()){
    kitap.setYayimTarihi(basta);
    doc.setString("basim_tarihi", kitap.getYayimTarihi());
}
if(!orijinal.isEmpty()){
    kitap.setOrijinalDili(orijinal);
    doc.setString("orijinal_dil", kitap.getOrijinalDili());
}
if(!acik.isEmpty()){
    kitap.setAciklama(acik);
    doc.setString("aciklama", kitap.getAciklama());
}
```

MutableDocument oluşturularak direkt döküman yapısında bilgileri depolayıp veritabanına yollayacağız. Gelen dolu alan bilgileri doc nesnesine ekleniyor ve aslında ilgili kitabın bilgi yapısı oluşturulmuş oluyor.

```
try {
    Objects.requireNonNull(database.getDefaultCollection()).save(doc);
    Toast.makeText(context: KitapEkle.this, text: "Kitap başarıyla yerel alana kaydedildi", Toast.LENGTH_SHORT).show();
} catch (CouchbaseLiteException e) {
    e.printStackTrace();
    Toast.makeText(context: KitapEkle.this, text: "Kitap yerel alana kaydedilemedi", Toast.LENGTH_SHORT).show();
    return;
}

try {
    URI syncGatewayURI = new URI("ws://10.0.2.2:4984/booksdb");
    ReplicatorConfiguration config = new ReplicatorConfiguration(database, new URLEndpoint(syncGatewayURI));
    config.setReplicatorType(ReplicatorConfiguration.ReplicatorType.PUSH_AND_PULL);
    config.setContinuous(true);
    config.setAuthenticator(new BasicAuthenticator("username": "kitap", "password": "kulsifre.toCharArray()));

    Replicator replicator = new Replicator(config);
    replicator.start();
    Toast.makeText(context: KitapEkle.this, text: "Veriler yükleniyor...", Toast.LENGTH_SHORT).show();
    startActivity(new Intent(context: KitapEkle.this, MainActivity.class));
} catch (URISyntaxException e) {
    e.printStackTrace();
    Toast.makeText(context: KitapEkle.this, text: "Veri tabanı ile bağlantı problemi oluştu.", Toast.LENGTH_SHORT).show();
}
```

Sonrasında Database nesnesi oluşturuluyor. Bu nesne aslında mobil cihaz üzerinde yerel depolama yapmayı sağlıyor. Bunun üzerinde verileri depolayıp sync işleminin gerçekleştirildiği _default collectiona ilgili oluşturulmuş bu dökümanı ekliyoruz.

Sync işleminde bu database eklenmesinin yapılabilmesi için sync gatewaye replicator ile işlem parçacığı oluşturmak gerekiyor. Gerekli bilgiler config ile replicatöre eklenip gatewaye bağlanmaya çalışılıyor. Bağlanınca da database gatewaye eklenmiş oluyor.

	d76261f4-a9ca-404f-b80a-6c08d02bee27	({"yazar":"test yazarı","kategori":"test","type":"book","baslik":"test kitabı"})	Windows'u Etkinleştir
	dd8bd8e5-e795-4d11-8ebf-7783dc7b92af	({"yazar":"J.R.R.Tolkien","kategori":"Fantastik","type":"book","baslik":"Hobbit","orijinal_dil":"ingilizce"})	İleştirmek

Main sayfaya girildiğinde veritabanındaki kitap bilgilerinin gözükebildiğini söylemiştik. Bu bilgileri veritabanından çekme esnasında sırasıyla,

```
1 usage
public List<Kitap> kitapBilgileriniAl() throws IOException{
    OkHttpClient client = new OkHttpClient();
    Request request = new Request.Builder()
        .url("http://10.0.2.2:4985/booksdb/_design/books/_view/byType")
        .build();

    try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }

        String responseBody = response.body().string();
        return bilgileriAyrıştır(responseBody);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

veri tabanına bağlanıyor. Bu esnada döküman içerisinde type: books şeklinde kaydettiğimiz için bu işlemi kolay gerçekleştirmek için bir view oluşturup bunu yayınlıyoruz. Yayınladıktan sonra içerisindeki yapıya göre arama yapmış oluyor.

The screenshot shows the Apache CouchDB web interface. On the left is a sidebar with navigation links: Dashboard, Servers, Buckets, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, and Views. The main area is titled 'Sample Document: _sync:dcpc_k::SGI-v-3.1-commit--uid-89a25a5e-1c31-11ef-853a-6c24086c51fe5'. It displays a JSON document with fields like 'DCPMeta', '1005', 'VbUID', 'StartSeqNo', 'EndSeqNo', 'SnapStartSeqNo', 'SnapEndSeqNo', 'FailoverEntries', and '101'. Below the document is a 'View Index Code' section with a 'Map' function:

```
1 function(doc, meta) {
2   if (doc.type == 'book') {
3     emit(meta.id, doc);
4   }
5 }
```

 A text box with the text 'İçerisinde filtreleme yapan yapı burası' points to the 'if' condition in the Map function. To the right of the Map function is a 'Reduce' section with a built-in function:

```
1 (built in: _count, _sum, _stats)
```

 At the top right of the main area are buttons for 'Load Another Document' and 'Edit Document'. Below the document and Map/Reduce sections are buttons for 'Make Copy' and 'Save Changes'. At the bottom of the interface, there is a 'books-bucket' dropdown, a 'Development Views' button, and a 'Production Views' button with a red notification icon. Below these are buttons for 'Delete', 'Compact', 'Add View', and 'Publish'. At the very bottom, there is a 'byType' dropdown and buttons for 'Delete' and 'Edit'.

Bu yapının çalıştırılmasıyla elde edilen sonuçlar cloud serverda böyle de gösterilebiliyor.

Results

Development Time Subset

Full Cluster Data Set

<

>

Show Results

Key	Value
"32314230-4a47-4374-97db-bcd553155ca8" 32314230-4a47-4374-97db-bcd553155ca8	{ "yazar": "test yazarı", "kategori": "test", "type": "book", "baslik": "test kitabı" }
"3bb94d4b-7d41-4073-ac51-711c3c61c9af" 3bb94d4b-7d41-4073-ac51-711c3c61c9af	{ "yazar": "test yazarı", "kategori": "test", "type": "book", "baslik": "test kitabı" }
"9b695e2b-6349-498c-b94f-4f5a04d2dc6b" 9b695e2b-6349-498c-b94f-4f5a04d2dc6b	{ "yazar": "test yazarı", "kategori": "test", "type": "book", "baslik": "deneme", "orijinal_dil": "turkce" }

Windows'u Etkinleştir

```
private List<Kitap> bilgileriAyrıştır(String json) throws IOException {  
    List<Kitap> kitaplar = new ArrayList<>();  
  
    try {  
        JSONObject jsonObject = new JSONObject(json);  
        JSONArray rows = jsonObject.getJSONArray( name: "rows");  
  
        for (int i = 0; i < rows.length(); i++) {  
            JSONObject row = rows.getJSONObject(i);  
            JSONObject doc = row.getJSONObject( name: "value");  
  
            if(doc.has( name: "yazar"))  
                yazar = doc.getString( name: "yazar");  
            else  
                yazar = "";  
            if(doc.has( name: "kategori"))  
                kategori = doc.getString( name: "kategori");  
            else  
                kategori = "";  
            if(doc.has( name: "baslik"))  
                baslik = doc.getString( name: "baslik");  
            else  
                baslik = "";  
            if(doc.has( name: "orijinal_dil"))  
                orijinaldil = doc.getString( name: "orijinal_dil");  
            else  
                orijinaldil = "";  
        }  
    }  
}
```

```

if(doc.has( name: "aciklama"))
    aciklama = doc.getString( name: "aciklama");
else
    aciklama = "";
if(doc.has( name: "basim_tarihi"))
    tarih = doc.getString( name: "basim_tarihi");
else
    tarih = "";

Kitap book = new Kitap(baslik, yazar, kategori, orijinaldil, tarih, aciklama);
kitaplar.add(book);

```

Gelen bilgiler JSON formatında olduğu için bunların parçalanması ve ilgili alanlara bölünmesi gerekiyor. Üstteki fonksiyon da bunu yapıyor..

```

1 usage
private void getir() {
    new Thread(() -> {
        try {
            List<Kitap> bookList = kitapBilgileriniAl();

            runOnUiThread(() -> {
                kitapadapt = new KitapAdapter(bookList);
                kitapliste.setAdapter(kitapadapt);
            });
        } catch (IOException e) {
            e.printStackTrace();
        }
    }).start();
}

```

Bu getirilen değerleri hali hazırda zaten bir oluşturduğumuz Kitap sınıfı nesnesine atanıp bu nesnelerden bir list oluşturulduğu için adaptera bu listi yollayıp başlatabiliyoruz.

Sadece bu atanan değerlerin gözükmesini istediğimiz için esnek görüntü oluşması adına adapter sınıfı içerisindeki `onBindViewHolder`'da gerekli ui öğelerinin kendisiyle alakalı bilgi olup olmamasına bağlı görünürlüğü ayarlanıyor.

```

@Override
public void onBindViewHolder(@NonNull BookViewHolder holder, int position) {
    Kitap kitap = kitapListesi.get(position);

    if(kitap.getBaslik() != null && !kitap.getBaslik().isEmpty()) {
        holder.basliktw.setVisibility(View.VISIBLE);
        holder.basliktw.setText(kitap.getBaslik());
    } else {
        holder.basliktw.setVisibility(View.GONE);
    }
    if(kitap.getYazar() != null && !kitap.getYazar().isEmpty()) {
        holder.yazartw.setVisibility(View.VISIBLE);
        holder.yazartw.setText(kitap.getYazar());
    } else {
        holder.yazartw.setVisibility(View.GONE);
    }
    if(kitap.getKategori() != null && !kitap.getKategori().isEmpty()) {
        holder.kategoritw.setVisibility(View.VISIBLE);
        holder.kategoritw.setText(kitap.getKategori());
    } else {
        holder.kategoritw.setVisibility(View.GONE);
    }
    if(kitap.getOrijinaldili() != null && !kitap.getOrijinaldili().isEmpty()) {
        holder.oridiltw.setVisibility(View.VISIBLE);
        holder.oridiltw.setText(kitap.getOrijinaldili());
    } else {
        holder.oridiltw.setVisibility(View.GONE);
    }
    if(kitap.getAciklama() != null && !kitap.getAciklama().isEmpty()) {
        holder.aciklamatw.setVisibility(View.VISIBLE);
        holder.aciklamatw.setText(kitap.getAciklama());
    }
}

```

DEMO FOTOĞRAFLARI

