**SUNUCU-İSTEMCİ SİSTEMİ TASARIMI**

Sistemimizin ihtiyacı doğrultusunda verilerin, seranın çalıştığı bilgisayardan bir ana sunucu bilgisayara yönlendirilmesi ve bu sunucu bilgisayarın seraların yakınsayacakları sıcaklıkları kontrol edebilmesi amaçlanmıştır.

Seraların her biri birbirinden farklı bilgisayarlarda çalışacağından dolayı her bilgisayarın öncelikle bir lokal ağ yapısı altında toplanması ilk yapılan işlemdir. Esasen farklı lokal ağlarda bulunan bilgisayarların birbirleriyle aynı lokal ağdaymış gibi haberleşebilmesi için Hamachi kullanılmıştır. Hamachi sanal bir lokal ağ oluşturmaya yarayan basit bir yazılımdır. Ardından sistem sunucu yazılımın bulunacağı bir ana bilgisayar ve istemci yazılımın bulunacağı sera bilgisayarları olarak ikiye ayrılmıştır. Platform olarak hem yönetim kolaylığı hem de bilgisayar, telefon, tablet gibi bir çok cihazda ortak çalışabilmesinden dolayı Web seçilmiştir.

**A) Sunucu Yazılımı:**

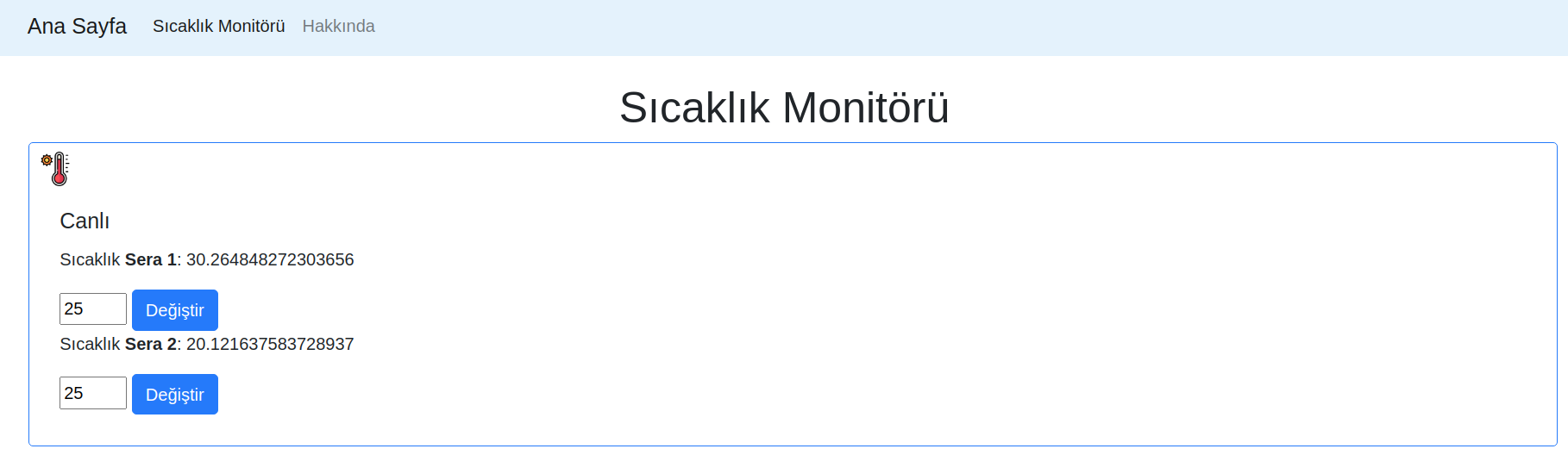
Sunucu tanım olarak, bir ağda diğer bilgisayarlar tarafından kullanıma/paylaşıma açık kaynakları barındıran bir bilgisayar demektir.

Bizim sunucu yazılımımız da yine aynı şekilde diğer bilgisayarı kontrol eder ve onlara hizmet sağlar. Sunucu yazılımımızın yaptığı işler şunlardır:

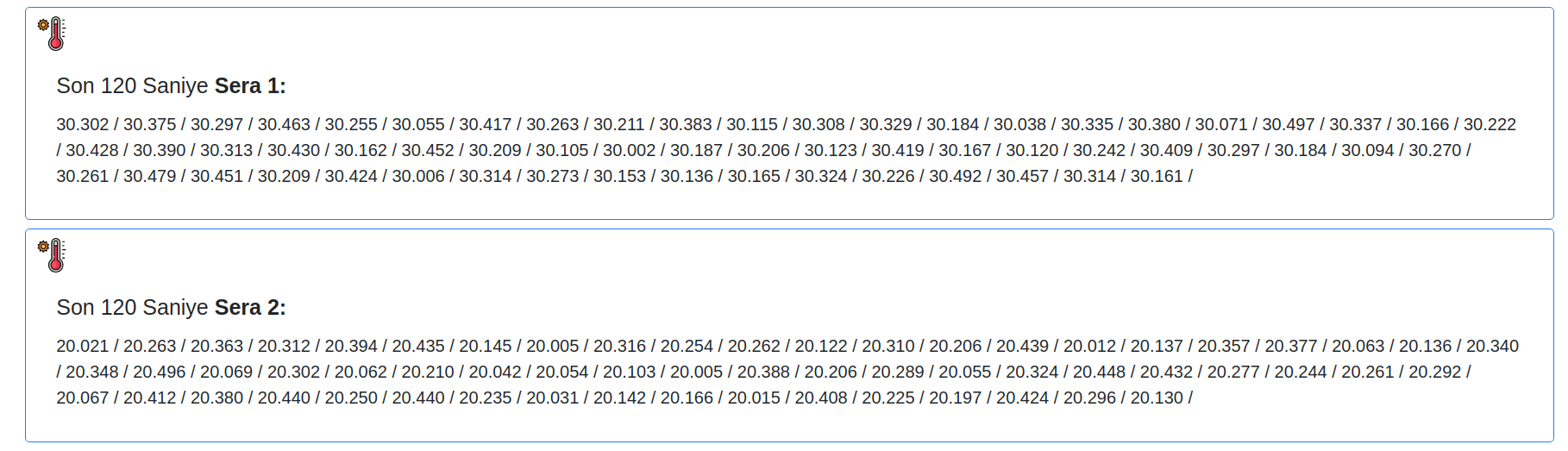
1. Ağda bulunan tüm seraların anlık ve geçmiş sıcaklık değerlerini görüntüler ve grafikselleştirir.
2. Ağda bulunan tüm seraların yakınsayacakları sıcaklık değerlerini sağladığı ekrandan güncelleyebilir.
3. Tüm seraların sıcaklık bilgilerini zaman bilgisiyle beraber bir veri tabanında saklar.

*Sunucu Tarafında Kullanılan Yazılımlar:*

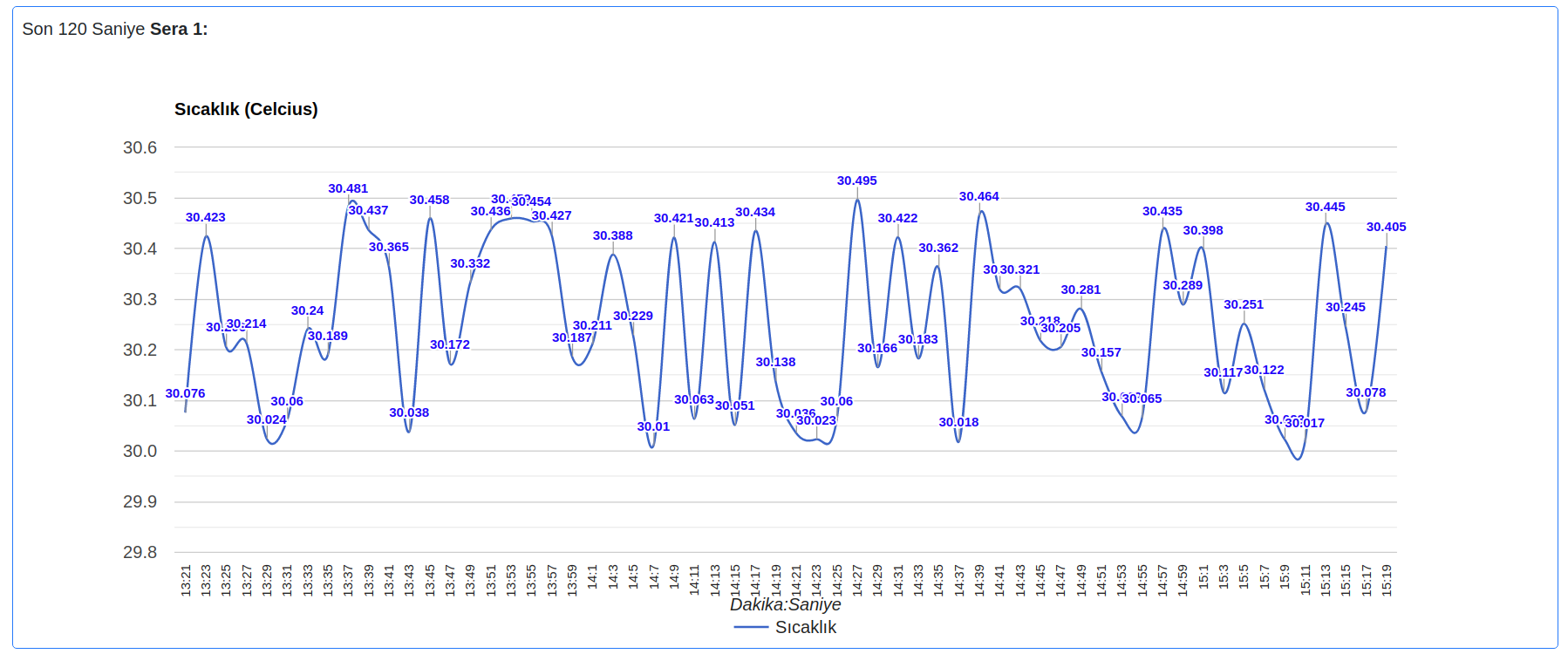
1. **Node.js:**Sunucunun Back-End tarafında bir Javascript çalıştırma ortamı olan, lokal bilgisayarımızda javascript kodunu çalıştırmamıza ve dosya sistemlerimize sıkıntısız erişmemize olanak saylayan Node.js kullanılmıştır. Node.js Joyent tarafından 2009 yılında geliştirilmeye başlanmış bir Javascript Runtime enviromentidir ve sunucu yazılımlarında oldukça popüler bir şekilde kullanılır.
2. **Express.js:** Express modülü Node.js tabanlı bir web uygulama sunucu çatısıdır. Node.js içerisindeki HTTP modülüyle route işlemlerini oldukça düzenli ve basit hale getirir.
3. **Pug:** Bir HTML preprocessorudur. Express.js ile beraber kullanılır.
4. **Bootstrap:** Web sayfalarını geliştirmek için kullanılabilecek bir önyüz çatısı ve araçlar bütünüdür. Twitter tarafından geliştirilmiş olup açık kaynak kodludur.
5. **Google Charts Api:** Grafikselleştirme işlemi için kullandığımız Google tarafından geliştirilmiş bir Apidir.
6. **Fetch Api:** Monitör sayfasının sunucunun içerisinden bilgi almak için sunucuya istek yapabilmesinde kullanılan bir apidir.
7. **Cors:** Cross Origin Resource Sharing dediğimiz farklı originler arasındaki haberleşmeye izin vermek için kullanılan bir kütüphanedir.

Figür 1.0

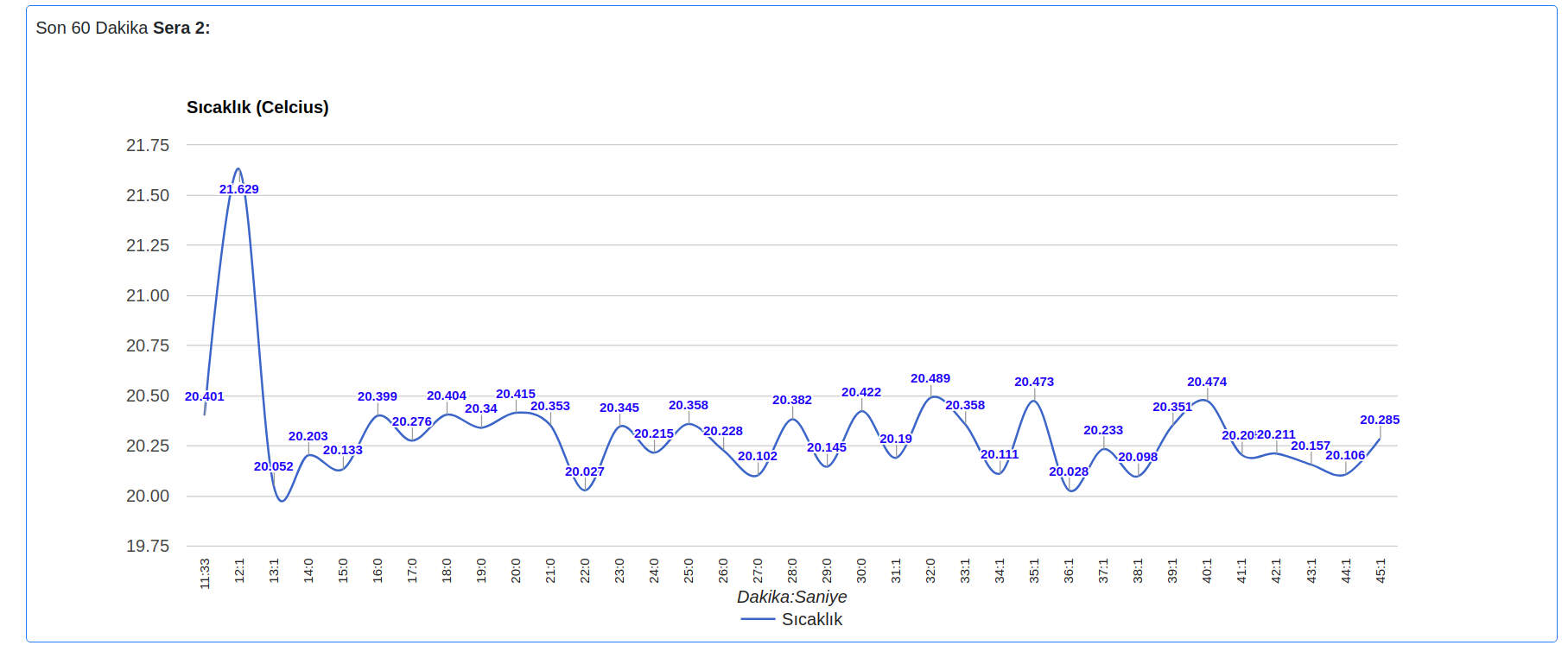
Yukarıda sunucunun sağladığı Monitör ekranını sayfasının en üst bölümü gözükmetedir. Burada farklı seralardan gelen sıcaklıklar anlık olarak gösterilir. Bunun yanında eğer yönetici kişi yakınsanan sıcaklık değerini değiştirmek isterse ilgili seranın altında bulunan input ve buton yardımı ile bu işlemini gerçekleştirebilir.

Figür 1.1

Yukarıda seraların son 120 saniye içerisindeki tüm sıcaklık verileri listelenmiş bir biçimde yöneticiye sunulmaktadır.

Figür 1.2

Yukarıda bir seranın sıcaklık bilgisinin son 120 saniye içerisindeki kayıtların grafikselleştirilmiş hali yer almaktadır.

Figür 1.3

Yukarıda bir seranın sıcaklık bilgisinin son 60 dakika içerisindeki kayıtları grafikselleştirilmiştir.

Tüm Figürlerde sayfa yenilense de açık bırakılmış olsa da değerler güncel kalmaktadır. Yani sayfa yenilendiğinde veriler, oluşturduğumuz veri tabanından zaman bilgilerine göre çekilerek yerlerine yerleştirilmektedir. Aynı zamanda sayfanın açık kalması durumunda da tüm içerik anlık olarak güncellenmektedir.

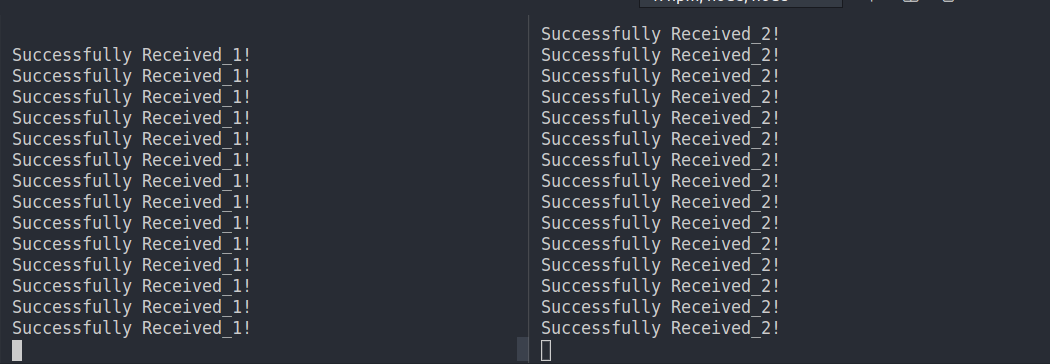
Sunucunun çalışmasındaki senaryo şöyledir. Sunucu her istemci için farklı path’den anlık olarak dinlemektedir. İstemcilerden veri gelmesi durumunda sunucu bu sıcaklık verisinin içerisine o verinin kendisine ulaştığı zamanı da ekleyerek bir sözlük haline getirir. Yani verimiz { \_id: zaman, temp: sicaklik } şeklinde bir json objesi olarak veri tabanına eklenmekte ve aynı zamanda monitör sayfasının erişebilmesi için farklı bir portta dinlenmektedir. Monitör sayfası bu sunucu verisine erişim için istek göndereceği zaman bu porta göndermektedir. Sonrasında sunucu verileri monitöre göndermektedir. Monitör her 2 saniyede bir istek gönderecek şekilde ayarlanmıştır. Yani 2 saniyede bir eğer sunucuda yeni veri var ise monitöre iletilecektir. Ayrıca monitör sayfası yeni yüklendiğinde her sera için son 120 saniye ve son 60 dakikanın sıcaklık verileri için de istek göndermektedir. Burada da iletişim yine aynı mantık ile sağlanmaktadır.

**B) İstemci Yazılımı:**

İstemci sunucuya erişen, ona veri gönderen ve ondan hizmet alan bir bilgisayar yazılımıdır.

İstemci yazılımımızın yaptığı işler şunlardır:

1. Seranın anlık sıcaklık bilgisini seri port haberleşmesi ile elde eder.
2. Anlık sıcaklık bilgisini her 2 saniyede bir sunucuya gönderir.
3. Sunucudan gelen verinin başarıyla ulaştığına dair mesaji konsola log eder.

Figür 1.4

Sunucuya başarıyla ulaşan veri sonucunda yukarıdakini kendi konsoluna yazar. Solda gördüğünüz sera-1 sunucunun dönütü, sağdaki ise sera-2 için sunucunun dönütüdür.

*İstemci Tarafında Kullanılan Yazılımlar:*

*(Sunucudakiyle aynı amaçlar doğrultusunda kullanılmışlardır.)*

1. **Node.js**
2. **Fetch Api**
3. **Express.js**
4. **Cors**

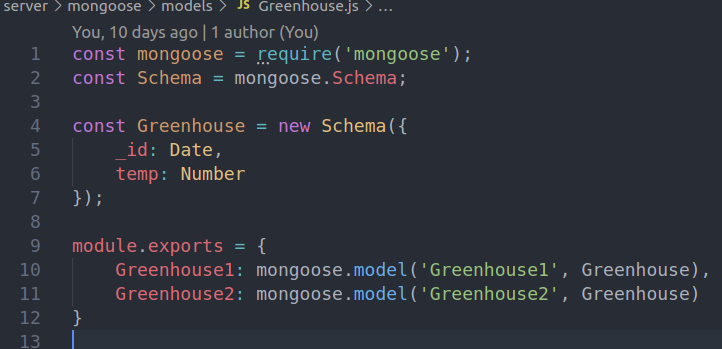
Kullanılan yazılımlardan anlaşılacağı üzere istemci tarafı sunucuya göre çok daha az kütüphane, api vs. ihtiyaç duymuştur. Yalnızca sunucudan hizmet alabilmesi için gerekli olan yazılımlara ihtiyaç duyar.

**C) Veri Tabanı:**

Veri tabanı isminden de anlaşılacağı üzere verinin saklanmasını ve tekrar tekrar kullanılmasını sağlayan bir yapıdır. Verinin kalıcılığını sağlar.

Veri tabanı yapısını sistemimize entegre ederken bir çok seçenek vardı ama bizim ihtiyacımızı en iyi karşılayacak ve kullandığımız yazılımlarla uyumlu çalışan bir yapısı olan MongoDB veri tabanı yönetim sistemini tercih ettik. MongoDB NoSQL olarak sınıflandırılan, ilişkisiz bir veri tabanı yapısıdır. Yani SQL veri tabanlarında olduğu gibi tablolar ve tablolar arası ilişkiler yoktur. Bu da bizim işimize gelmektedir. Çünkü sadece sıcaklık ve zaman bilgisi tutacağımız için SQL yerine NoSQL bir yapı kullanmak veriye erişim hızını ciddi ölçüde artırmaktadır.

MongoDB’nin Node.js’te kullanımı yine Node.js’in bir modülü olan Mongoose ile oldukça kolaya indirgenmektedir. Mongoose MongoDb işlemlerini kolaylaştıran bir ODM modülüdür.

Figür 1.5

Yukarıdaki figürde gösterilen bizim tek sera modelimizdir. Her sera için veri yapısı aynı olacağı için tek model yeterli olmaktadır. ‘\_id’ değeri her sıcaklık verisinin özel (unique) değeri olacağı için ‘\_id’ olarak isimlendirilmiştir ve verinin kaydedildiği zaman bilgisini tutmaktadır. ‘temp’ değeri ise verinin sıcaklık değeridir. En alt kısımda ise aynı model farklı isimler ile export edilmiş ve sunucu içerisinde kullanımı sağlanmıştır.