在这个实验室中，我们将创建一个RESTful API作为MongoDB数据库的DBMS，这是我们在上一个实验室中创建的。

在那个API中，我们将指定DBMS的端点和行为，它们将规范我们如何定义数据结构，以便将所有东西连接在一起。

我们旅程的最后一步将包括修改实验室6中创建的Notes应用程序，这样我们的Notes将在线保存和检索!

第1部分—需求设置

作为第一步，我们将克隆包含服务器的GitHub项目到桌面上的一个文件夹中。

注意，在第8行中，我们设置了start命令，以便Zeit在index.js文件上运行node，我们还没有这个文件，但是很快就会有了。

第13行设置要安装的依赖项，Zeit将自动执行该安装，但是只有在我们在配置文件中指定它的情况下，我们现在就这样做!

现在，我们可以将该文件保存在Repo的文件夹中，并将其命名为“now.json”。

在第3行中，我们指定了项目的名称。

在第4行到第12行中，我们指定了如何构建“索引”。js”和“包。json "文件，我们使用node来实现。

在第13行中，我们指定了可以访问我们的服务器的路由，以及它们应该被指向的位置:在本例中，所有路由都应该在“/index”中转移。当然，这是我们的服务器。

到目前为止，一切顺利，现在让我们在Sublime中创建一个新文件，并将其保存为index.js，再次保存在Repo的文件夹中。

我们将使用这个文件作为我们的服务器，通过以下步骤:

▪导入所有必要的库

▪使用默认返回到“/hi”端点来测试路由

▪建立到MongoDB数据库的连接

▪为下列操作创建处理程序:

o收集所有笔记

o创建一个新音符

o得到一个明确的说明

让我们从导入index.js文件中必要的库开始:

完成这一步后，让我们创建一个测试路径，并导出我们的应用程序，让Zeit使用:

现在，我们可以将更改提交到GitHub并在线访问我们的服务器，以测试此路由是否工作!

进入你的Zeit仪表盘，点击你的服务器来检查它是否运行，

它应该会显示以下屏幕:

这是因为我们没有“/”路由的处理程序，但是如果我们在末尾添加“/hi”，我们将看到:

哇,这工作!

让我们通过添加剩余的API端点和处理程序来完成这个实验。完成这一步后，我们将使用自定义GET和POST请求测试数据库的功能，这些请求将在我们的应用程序中用于从数据库收集数据!