



后端开发岗就业形势与学习路径

|  |  |
| --- | --- |
| **专 业** | **软件工程** |
| **姓 名** | **孙明志** |
| **日 期** | **2022/8/30** |

**摘 要**

毋庸置疑的是,在疫情和全球经济形势下行的大环境下,一份称心如意的工作已经愈发难寻,如此便要求我们未雨绸缪.本文将简要介绍后端的各个岗位要求,准备的技术栈以及对应的学习路径,开发岗薪资比不上算法岗,但市场需求相对较多,不失为一种好选择.

关键词:开发岗 后端 学习路径

**目 录**

[**第1章 行业概述与应届生就业现状** - 1 -](#_Toc113484521)

[1.1 后端开发究竟是什么？ - 1 -](#_Toc113484522)

[1.2 后端开发人员做什么？ - 1 -](#_Toc113484523)

[1.3就业形势 - 1 -](#_Toc113484524)

[**第2章 职位概览** - 2 -](#_Toc113484525)

[2.1 互联网公司的技术部门组织架构 - 2 -](#_Toc113484528)

[2.2.1从单体到分布式：后端工程师的分工 - 2 -](#_Toc113484529)

[2.2.2 中间件/数据库：后端服务的基石 - 2 -](#_Toc113484530)

[2.2.3数据平台：公司决策的引擎 - 3 -](#_Toc113484531)

[2.2.4 运维/安全：保证服务的高可用和安全性 - 3 -](#_Toc113484532)

[2.2.5 一个大型科技公司的技术部门组织架构：从技术部到中台 - 3 -](#_Toc113484533)

[2.2 职位概览 - 4 -](#_Toc113484534)

[**第3章 需要怎样的技术栈及学习路径** - 6 -](#_Toc113484535)

[3.1需要怎样的技术栈? - 6 -](#_Toc113484536)

[3.2学习路径 - 6 -](#_Toc113484537)

[3.2.1计算机考研 408 基础 - 6 -](#_Toc113484538)

[3.2.2算法 - 6 -](#_Toc113484539)

[3.2.3后端语言基础 - 7 -](#_Toc113484540)

[3.2.4 MySQL - 7 -](#_Toc113484541)

[3.2.5 Redis - 8 -](#_Toc113484542)

[**结 论** - 9 -](#_Toc113484543)

[**参考文献** - 10 -](#_Toc113484544)

**第1章 行业概述与应届生就业现状**

## 后端开发究竟是什么？

软件应用程序就像冰山一样。用户看到的只是应用程序的一部分——在大多数情况下——应用程序的最大部分是看不到的。这就是令人难以捉摸又神秘的“后端”

在Web开发的篇章中，我们主要讨论Web开发，因为它涉及与最终用户的直接互动——我们可以称之为“前端Web开发”。在大多数应用中，非用户接口代码比用户接口代码多得多。复杂系统是在后台发生的各种使其工作的逻辑。

我们需要存储和检索数据，需要遵循业务逻辑和规则，并且需要对结果进行预测。所有这一切都发生在幕后。

简单来说，后端开发就是开发人员编写的不能直接看到的代码。

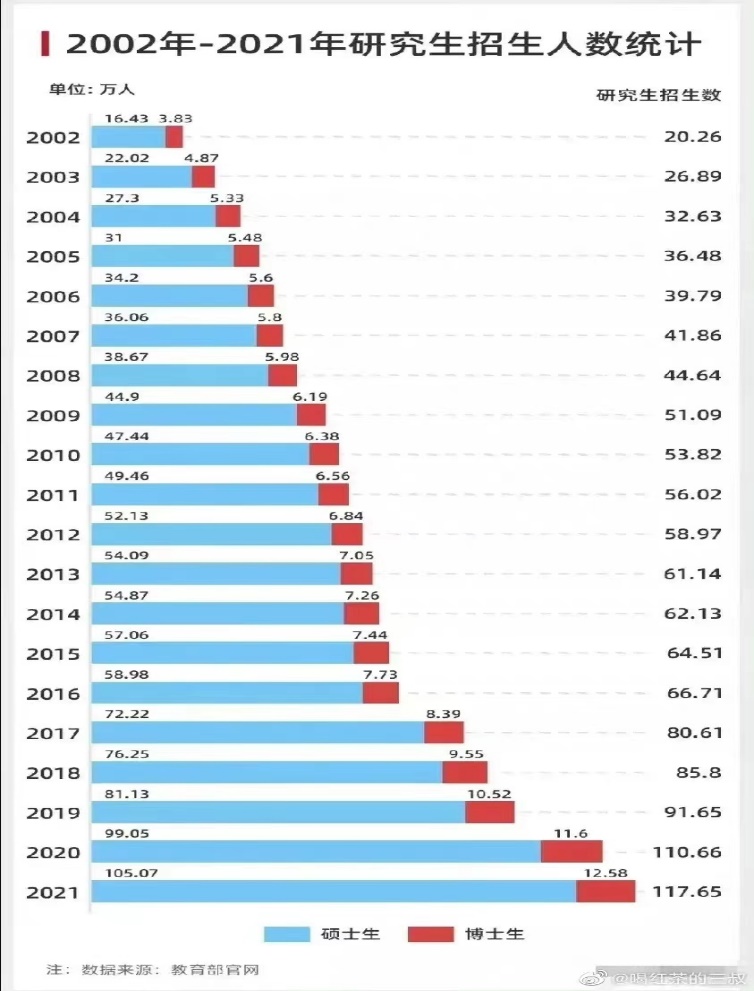
## 1.2 后端开发人员做什么？

根据正在处理的应用程序的大小和范围，后端开发人员要做的事情有很大的不同。在我是一个后端开发人员时，我做了很多工作，工作于应用程序中的业务逻辑，以及从前端提取和检索数据。

　　在Web开发世界中，大多数后端开发人员从事于构建他们正在工作的应用程序背后的实际逻辑。

　　后端开发人员也可能会大量参与系统架构，决定如何组织系统的逻辑，以便能够正常维护和运行。

他可能会参与构建框架或系统架构，以便于更容易编写程序。后端开发人员比前端开发人员花费更多的时间在实现算法和解决问题上。



## 1.3就业形势

无须讳言,在后疫情时代,经济下行,加之每年越来越多的毕业生,互联网行业的泡沫已经渐渐消退,一方面是越来越多的毕业生以及转码的求职者,一方面是经济下行下各企业也在缩招裁员,就业压力也是越发加大,往年毕业生保底的选择已经被卷成了一片红海.在大环境无法改变的情况下,这就更加要求我们磨炼专业技能.宽慰地说,在如今的经济形势下,互联网仍然是相对而言不错的选择.

**第2章 职位概览**



## 2.1 互联网公司的技术部门组织架构

### 2.2.1从单体到分布式：后端工程师的分工

要理解一个（大型）互联网公司的组织架构，首先要理解一个产品的后端服务从 0 到 1 的发展过程。

大致来说就是这样的过程：

单机系统：Java/PHP/Node/Python + Linux + MySql + Nginx。 ==>

流量增大，数据库读写出现瓶颈，数据库分库分表 ==>

流量继续增大，单机部署无法应对高并发，于是集群部署，横向拓展 ==>

为了提高性能，抽出单独的缓存服务 ==>

分布式数据库/分布式存储 ==>

微服务架构 ==>

最终进化为适应于特定业务场景的复杂分布式架构

在小型的公司，一般就只有一个后端开发团队，里面没有非常明确的角色分配。服务，数据库，缓存，部署，都要自己来搞。当公司变大的时候，就会出现一些分工。

开发工程师主要负责开发服务，运维和 DBA 负责服务的部署，数据库的安全和效率。架构师负责设计整个应用的架构。

### 2.2.2 中间件/数据库：后端服务的基石

上一节说到互联网公司的后端架构最终演化为一个分布式的系统。那一个分布式的系统运行，需要许多中间件的支持。比如消息队列，分布式缓存，分布式数据库，微服务框架，容器编排调度等等。这些被其他服务调用的，负责通信/存储的服务，就是所谓的中间件。中间件可以说是连接两个服务的一个中介。

公司越来越大，就需要有人专门去维护这些公共组件，让开发工程师可以专注于业务逻辑的开发。比如维护自己的消息队列，数据库，缓存，RPC框架等等。在大公司中，开源的方案并不能满足业务需求，所以需要对开源的方案进行改进和创新。

上一节中的分工里没有提到的，就是中间件研发的工程师。这些工程师在一个大型的互联网公司中是非常重要的。

### 2.2.3数据平台：公司决策的引擎

科技公司在运行中会产生各种各样的数据，比如用户行为数据，订单数据，访问记录数据。还有服务器的运行状态等等数据。这些数据都是一种资产，是公司来了解业务情况的一个窗口。

大数据这个方向其实已经非常成熟了，它主要包括数据的搜集，清洗，加工，存储，计算，分析。在这个链路上，需要很多工程师的努力。我们需要工程师来进行数据的清洗和加工，需要工程师来搭建数据仓库，数据加工平台，数据分析平台。需要工程师来优化数据实时和离线计算的引擎。需要工程师来写各个终端的数据上报服务。

所以大型的互联网公司都会有一个数据部门，让数据产生价值。

### 2.2.4 运维/安全：保证服务的高可用和安全性

传统意义上的运维负责公司服务器的管理，生产环境的稳定性，公司机房的设计和日常维护。大型公司中，还会涉及异地多活等等高可用方案。

运维这个角色现在也叫运维开发（DevOps）或者 SRE（网站稳定性工程师）。通过脚本和工具进行自动化的运维。比如开发要使用数据库，运维就搭建一个多副本的数据库集群，并且让数据库可以自动备份，还写了一个平台来访问和管理数据库。又或者有一个基础软件，需要安装在 500 台机器上。运维就开发了一个软件，可以让大家一键部署。

安全则是一个很大的领域，主要负责产品的信息安全。公司里一般都会有安全部。

### 2.2.5 一个大型科技公司的技术部门组织架构：从技术部到中台

小型公司一般只有一个技术部，里面分前端后台等角色。大公司一般是按业务来划分部门。比如电商业务，金融业务，零售业务等等。所谓的业务最终会落地到一个实际的产品中，比如淘宝。一个业务部门可能会同时运营多款产品。

这里需要引入一个概念，叫中台。和中台相对的就是前台。前台部门就是业务部门，直接开发面向消费者的产品的。中台部门就是负责给前台部门输送弹药的。负责支持前台部门。这里的支持不是帮前台部门写业务代码，而是提供一些基础的服务。比如前文里说到的中间件和数据库，还有数据平台。

我们在写产品的时候，会有一个抽象公共代码的过程。把多个模块中公共的部分抽出来。那在公司的运营中，我们也可以把公共的职能抽出来，变为中台。

每个业务部门自己去维护自己的中间件或者数据库，或者搭建数据平台，会造成很大的资源浪费。对部门之前的合作也不利。一个科技公司，统一自己内部的技术栈和基础设施，有几点好处：

o 方便内部的技术交流和合作

o 营造统一的技术氛围，利于公司内部的认同感

o 方便对外输出技术

o 降低各部门开发的成本

其实中台战略的好处是显而易见的，让专业的人做专业的事，前台负责业务上攻城略地，中台部门负责中台能力的搭建，支持前台业务。

## 2.2 职位概览

**研发（Java/C++)**

支持各种系统（面向消费者的，或者内部系统）的研发。基础平台，人工智能，数据平台也需要有平台来展现，所以也需要研发工程师。

简单的说就是写接口的。

**数据研发**

企业的数据中台，从数据的采集，加工，计算，存储，分析，都需要熟悉大数据相关技术（数据仓库/实时计算/离线计算/数据清洗）的工程师来参与。

**基础平台研发（Java/C++/Go)**

数据库开发，网络（CDN，机房网络），分布式存储，云计算（虚拟化，操作系统），云计算网络（SDN），中间件（消息队列，缓存，分布式服务框架）

**运维（DevOps, DBA）**

运维开发，负责产品，数据库，中间件的运维，开发相应的软件来做运维自动化。也会做一些监控系统来监控服务的运行情况。

**算法**

算法其实不属于后端，是单独的一个类别，里面有视觉/语音/推荐等等各种小方向的职位。

**应用 vs 底层**

这么多后端方向的职位，我们可以大致分为**应用**和**底层**两个部分。

应用就是指业务代码，也就是所谓的写 API。

底层就是指中台下面的几个方向，比如中间件/数据平台等等的研发。当然底层里面也有数据库开发/Linux 内核开发等等偏**系统级软件**的开发。也有**平台软件/中间件**的研发，比如 K8s 这样的容器调度系统，或者是一个消息队列。这样的软件**更靠近应用层**，对底层操作系统的依赖没那么大。

今后去工作的时候也会遇到应用 vs 底层的选择。能直接投相关的职位当然是最好，大公司一般都会分清楚招的职位具体是做什么。有的公司可能是先以应用研发的名义统一招人，后面才会有具体的方向。先进入这个行业工作，然后选择自己的方向，也是一个不错的道路。

**第3章 需要怎样的技术栈及学习路径**

## 3.1需要怎样的技术栈?

应用研发的需求自然要大于底层研发，也只有比较大的企业才会有专职的底层研发。所以说如果技术栈和当前公司流行的技术栈匹配的话，就比较好找工作。现在比较流行的后端开发语言就是 Java 和 C++ 了。电商公司一般都是 Java。腾讯和百度是用的 C++ 比较多。微博用的是 Java 和 PHP。用 PHP 的大公司也有，但在公司里不是主流，所以 PHP 就可以排除了。同样 Python 在大公司里主要还是做一些辅助的工作，比如要开发一个运维系统，就可以用 Python 来写，这样比较快。

一些创业公司也喜欢用 Python 来快速搭建产品原型。比如豆瓣/知乎/字节跳动。但后期如果业务复杂了，一般会用 Java 重构。

Go 语言在大公司用的也挺多，主要是做云计算方向。但很少有大公司直接招 Go 工程师，所以首先精通 Java/C++ 的一种，然后还懂 Go，是比较好的一种选择。

底层研发用的语言就比较受限制了。中间件一般是 Java/C++，系统软件一般是 C/C++。云计算相关的底层系统，Go 用的会比较多。其他语言在底层开发里是很少见到的。

## 3.2学习路径

**简单总结**：

计算机考研 408 基础+算法题+一门后端语言+MySQL+Redis(+有开源经历更佳)

### 3.2.1计算机考研 408 基础

这部分就是我们平时的专业课,这部分可以针对每一门课，找到一门国外的课，跟着学完内容（PPT 和要求读的课本相关章节看过，理解，如果有课程视频要看视频），做完 Lab。作业可以根据自己兴趣来做。重点还是在 Lab。

这些 Lab 一般都是要你实现一些底层的东西，比如数据库系统，会让你实现 Buffer Pool 和 B+ Tree。计算机网络让你实现一个完整的 Web Server，还有 TCP 协议等等。分布式系统一般要求实现简单的分布式 KV 存储，还有 Raft 等分布式一致性协议。这些东西都是一门学科最精髓的，而你手动实现过了，那你对这个东西的理解自然远远超过普通的刷题看书的学习方式。

### 3.2.2算法

这里就是面向面试刷题,做题很有必要，理解一道题更有必要，总结题目类型事半功倍。刷题的可以按照专题去做，例如有二叉树、图论、并查集、二分法、动态规划等专题，例如你只有学会了0/1背包问题才可以去做完全背包问题。如果短时间内在专题题目中练习的话，会加深对这个数据结构和算法的理解。

leetcode是一个非常多人学习算法的社区，可以说现在想要进互联网大厂，都是需要刷题巩固的。刷题的顺序比比皆是,这里不再赘叙.

### 3.2.3后端语言基础

这里需要深入学习一门语言,这也是无论什么职业的基础,以JAVA为例,在熟悉基础语法后,还要学习并发编程,虚拟机等知识. 如果是面试Java后端，虚拟机的知识真是必看，但是由于应届生基本上项目精要为0，所以面试官不会问道项目排查方面的内容，只会问一些比较基础的比如：说一下堆栈的区别什么是双亲委派模型说一下类装载的执行过程针对Java学习，不要只停留于表面的语法，要多看多做，切忌眼高手低。

### 3.2.4 MySQL

数据库的学习，离不开实践。先在自己的电脑上安装一个MySQL,可以参考：

[https://www.runoob.com/mysql/mysql-install.html](https://link.zhihu.com/?target=https%3A//www.runoob.com/mysql/mysql-install.html)

安装版本最好**高于5.7**，因为有些特性才能学习到。

如果想要安装一个可视化工具，推荐**navicat** [http://www.navicat.com.cn/](https://link.zhihu.com/?target=http%3A//www.navicat.com.cn/)

安装完毕后开始MySQL的学习，针对这个的学习我分成了三个部分：

* 快速看完一个体系视频
* 操作数据库，练习SQL语句
* 结合InnoDB特性学习

1.**看完一个系统性的视频或者书籍**是为了能够让你在短时间内快速学习这个中间件，了解整个中间件的脉络。  
<https://www.bilibili.com/video/BV1UQ4y1P7Xr?p=75>

2.**操作数据库，练习SQL语句，我建议你直接使用终端进行操作。** 有些人会说现在ORM这么成熟了还要学习SQL吗？当然，因为万丈高楼平地起，ORM底层也是操作SQL的，在性能排查上学习点SQL大有益处。explain语句怎么分析？死锁状态如何查看？索引如何建立？都要使用SQL。

InnoDB支持事务，支持行锁，已经是最常用的存储引擎。  
四大特性：  
**原子性**（Atomicity）： 事务是最小的执行单位，不允许分割。事务的原子性确保动作要么全部完成，要么完全不起作用；  
**一致性**（Consistency）： 执行事务前后，数据保持一致，多个事务对同一个数据读取的结果是相同的；  
**隔离性**（Isolation）： 并发访问数据库时，一个用户的事务不被其他事务所干扰，各并发事务之间数据库是独立的；  
**持久性**（Durability）： 一个事务被提交之后。它对数据库中数据的改变是持久的，即使数据库发生故障也不应该对其有任何影响。

3.**多看一些MySQL的基础知识**，MySQL的知识说实话非常多，但是它的体系结构非常明显，学习起来有章可循。在面试中经常会问道一些固定答案的题目比如：

* MySQL索引使用的数据结构是什么
* redo log和undo log之间有什么异同
* 如何利用bin log做主从复制

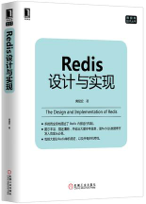
### 3.2.5 Redis

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的APIRedis的优势在于：性能极高 ：

Redis能读的速度是110000次/s,写的速度是81000次/s 。

丰富的数据类型 : Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。

原子 ： Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。学习redis推荐这本书 《Redis的设计与实现》



这本书深入浅出的描述了redis的几种数据结构的底层实现以及主从复制、事务性、持久化等特性，一句话总结就是非常全面地剖析，例如redis底层字符串是由SDS构成，压缩列表何时会失效等等。推荐一个学习网站： http://redisbook.com/

**结 论**

**道路是曲折的,前途是光明的.尽管经济下行就业前景低迷,但无可否认的是计算机相关岗位仍是工科专业中最为坚挺的,我们唯有充分发挥科班的优势,锻炼过硬本领,才能找到满意的岗位.**

**参考文献**

[1] 后端工作方向介绍[2018-09-14]. https://zxc0328.github.io/2018/09/14/backend-job-road/#more