实现轻协同IDE的技术选型经验

李亚飞

个人介绍

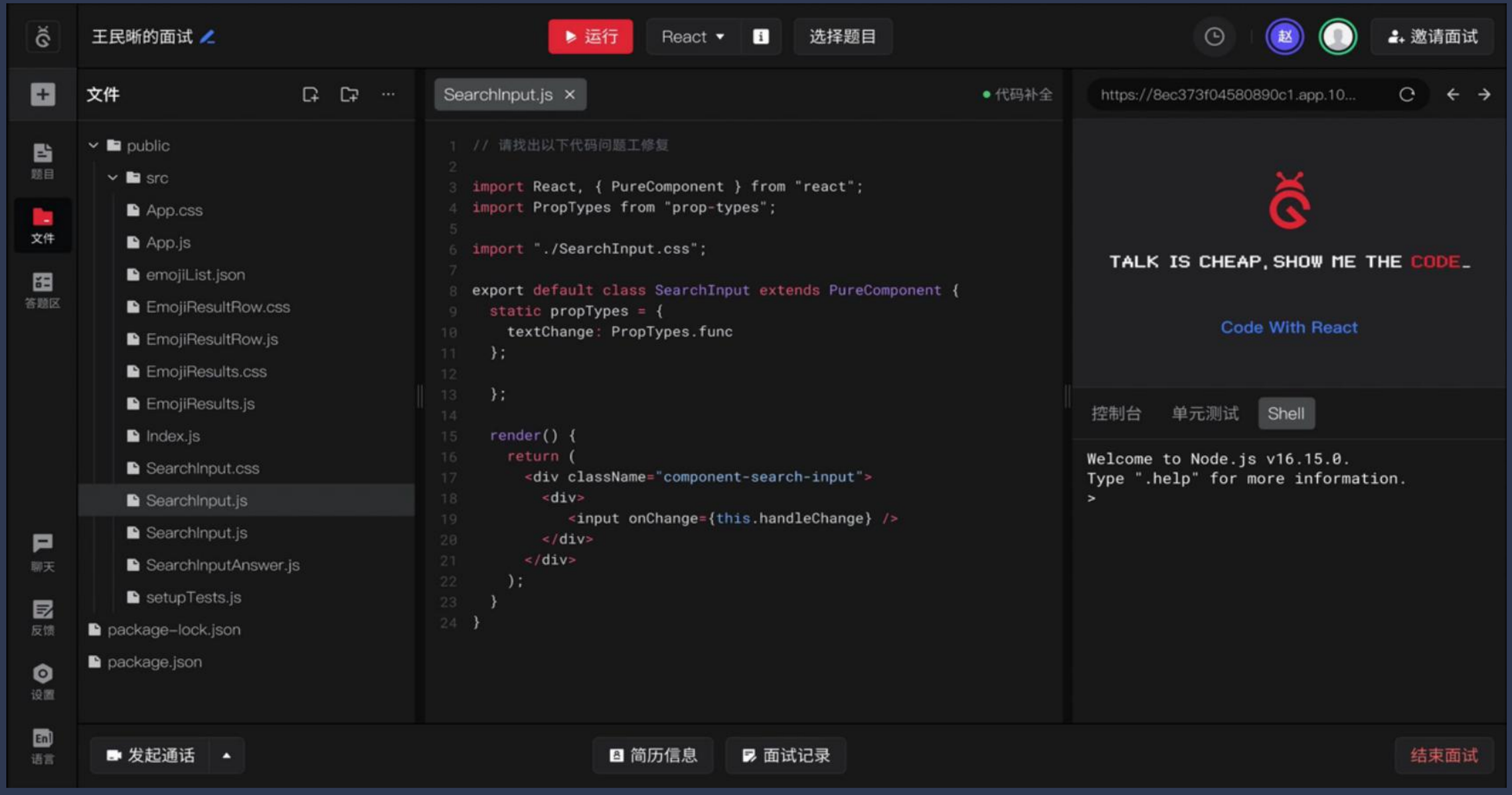
- 李亚飞，ShowMeBug CEO兼首席架构师
- 连续创业者，14年开始创业，9年创业经验
- 2009年毕业于吉林大学计算机系
- Ruby 社区核心贡献者，RubyConf China 三届讲师
- 前深信服资深架构师，自动化产品线主管&技术负责人
- 2015年，通过个人 Github 获得某前沿科技公司前端架构师职位
- 2021年科创中国深圳企业家 U30 上榜成员
- ShowMeBug 连续获得奇绩、盈动、变量、真格、红杉、高瓴六家近亿元融资

分享内容

- 轻协同IDE引擎的开发动机与要求
- 选择 PaaS 方案，提供 NPM 包的 SDK
- 选择 React + zustand 作为前端 SDK 组件
- 选型 socketio
- 协同层：NestJS + socketio
- 后端交互：MQ 消息队列，容器化机制

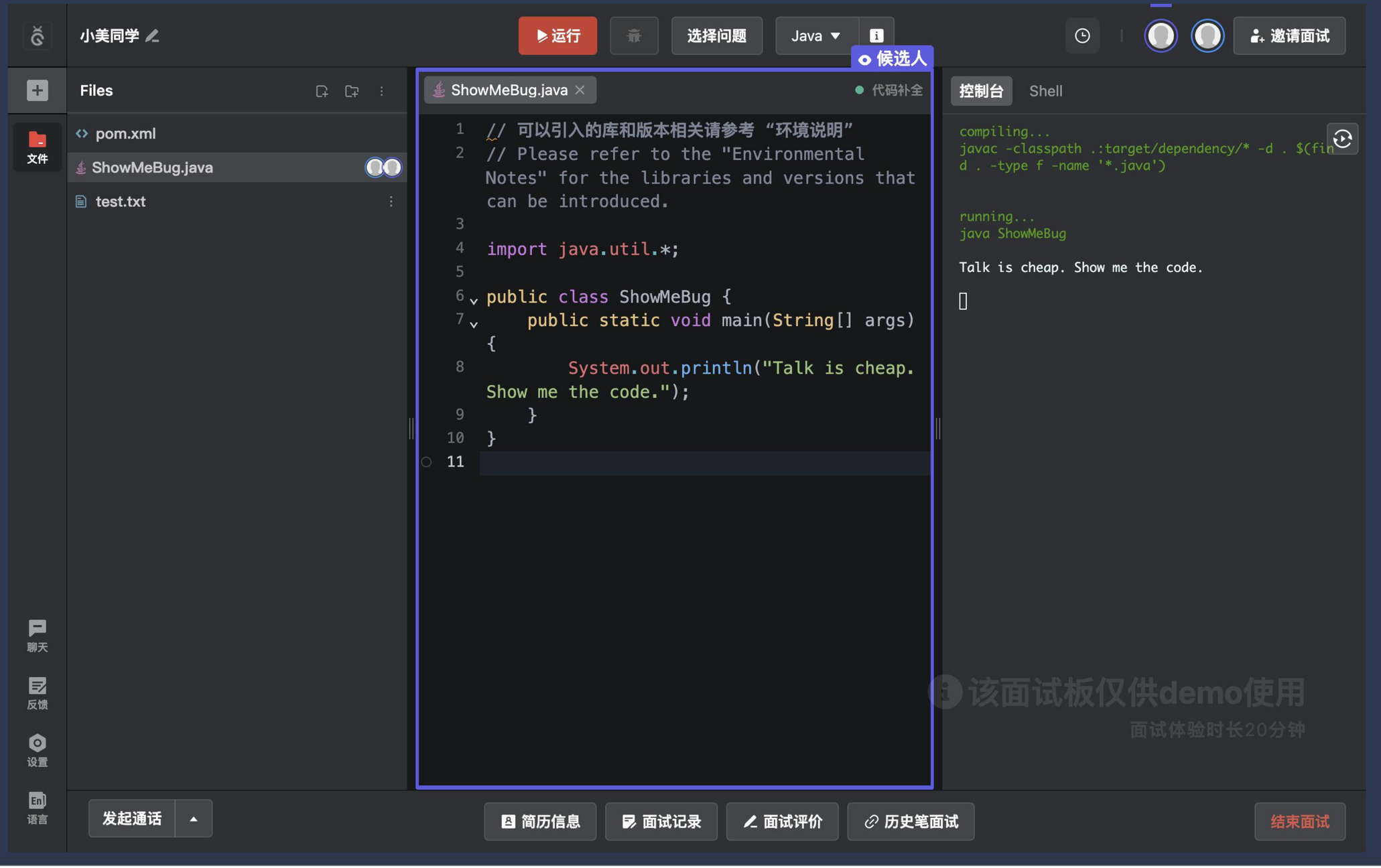
今天我们一起看看，NodeJS 全栈能支撑多大能力的业务?

轻协同IDE的前世今生


王民晰的面试
运行
React
选择题目
邀请面试
文件
SearchInput.js
代码补全
https://8ec373f04580890c1.app.10...
题目
文件
答题区
聊天
反馈
设置
语言
public
src
App.css
App.js
emojiList.json
EmojiResultRow.css
EmojiResultRow.js
EmojiResults.css
EmojiResults.js
Index.js
SearchInput.css
SearchInput.js
SearchInput.js
SearchInputAnswer.js
setupTests.js
package-lock.json
package.json
// 请找出以下代码问题工修复
import React, { PureComponent } from "react";
import PropTypes from "prop-types";
import "./SearchInput.css";
export default class SearchInput extends PureComponent {
static propTypes = {
textChange: PropTypes.func
};
};
render() {
return (

<input onChange={this.handleChange} />

);
}
}
TALK IS CHEAP, SHOW ME THE CODE_
Code With React
控制台
单元测试
Shell
Welcome to Node.js v16.15.0.
Type ".help" for more information.
>
发起通话
简历信息
面试记录
结束面试


小美同学
运行
选择问题
Java
候选人
邀请面试
Files
pom.xml
ShowMeBug.java
test.txt
文件
聊天
反馈
设置
语言
ShowMeBug.java
代码补全
1 // 可以引入的库和版本相关请参考“环境说明”
2 // Please refer to the "Environmental Notes" for the libraries and versions that can be introduced.
3
4 import java.util.*;
5
6 public class ShowMeBug {
7 public static void main(String[] args) {
8 System.out.println("Talk is cheap. Show me the code.");
9 }
10 }
11
控制台
Shell
compiling...
javac -classpath .:target/dependency/* -d . $(find . -type f -name '*.java')
running...
java ShowMeBug
Talk is cheap. Show me the code.
该面试板仅供demo使用
面试体验时长20分钟
发起通话
简历信息
面试记录
面试评价
历史笔面试
结束面试

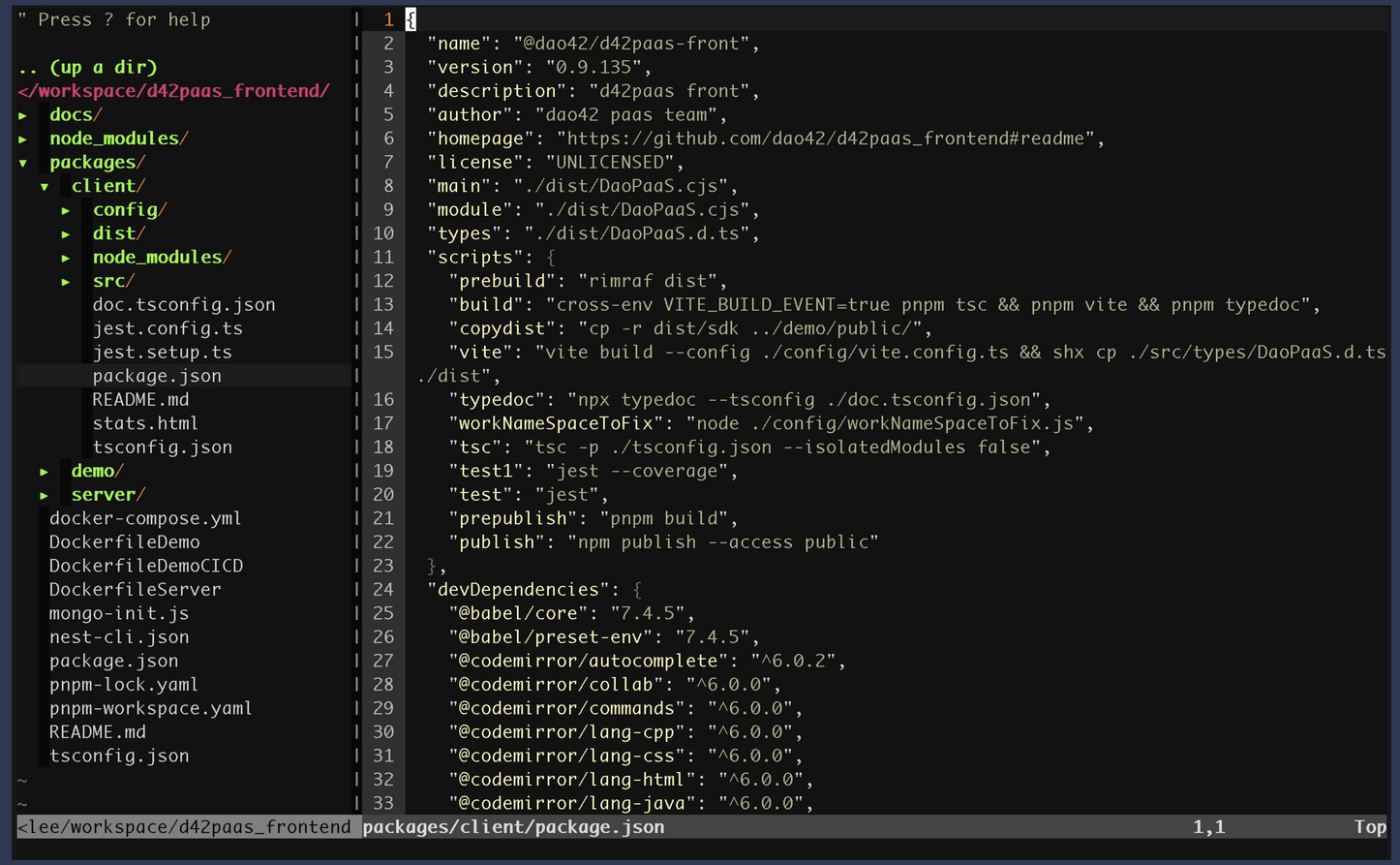
- 轻量级（简单，易用，免安装）
- 协同性（用于Coding面试、学习指导等场景）
- 零秒启动（极快的启动速度、所有包依赖提前准备）
- 零秒切换（秒换不同编程环境）
- 零秒延迟（所有操作实时进行，包括Shell、Console、文件树）

- 需要完全自主可控（排除 VSCode 定制方案）
- 需要超出预期的协同体验
- 需要超强自定义：答题区、隐藏锁定答案、测试用例解析等等
- 结论：排除 VSCode 等IDE定制方案，完全从零开发

接下来一起来进行架构之旅

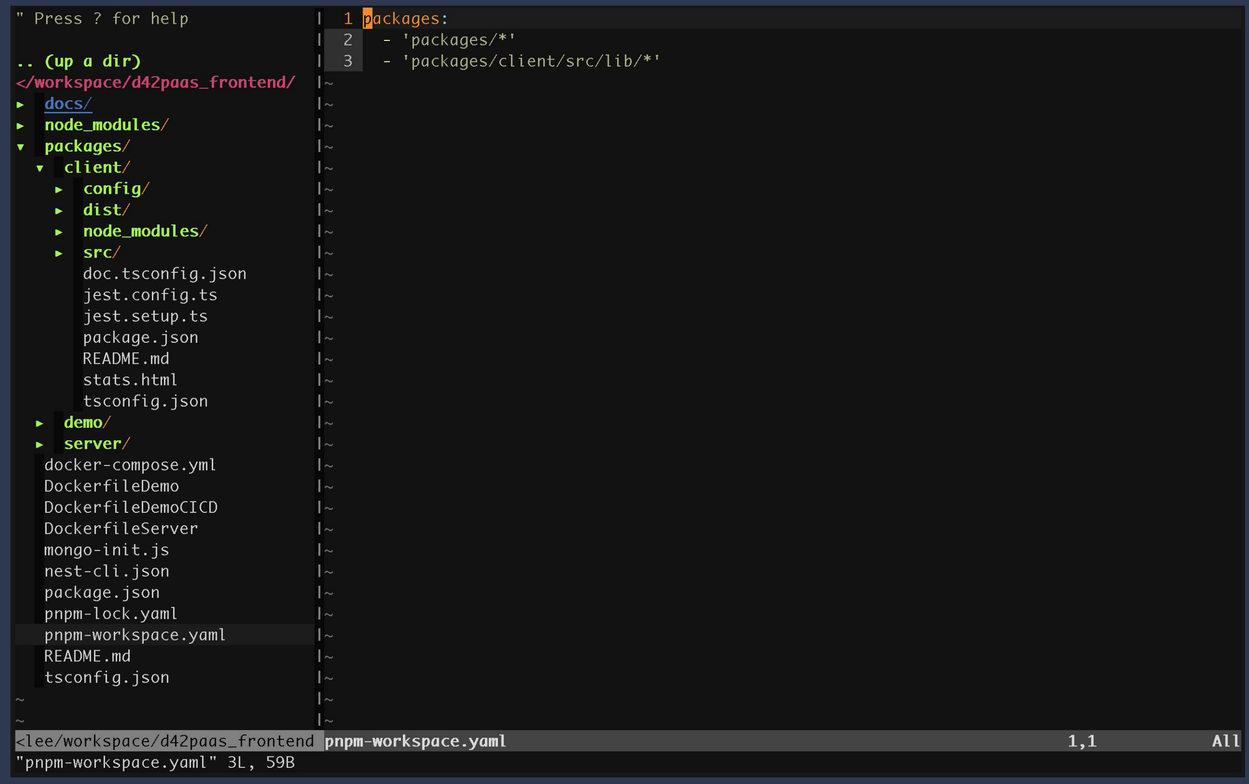
PaaS 模式

因为轻协同IDE拥有很多应用场景：技术测评、在线面试、编程学习，为了支持不同的业务，所以采用 PaaS 化的架构。NPM 是必然的采用的打包方案。



MonoRepo

- pnpm 提供更快的安装速度
- demo：提供演示和测试能力
- client：SDK核心代码
- server: Node层负责长连接协同



React + zustand

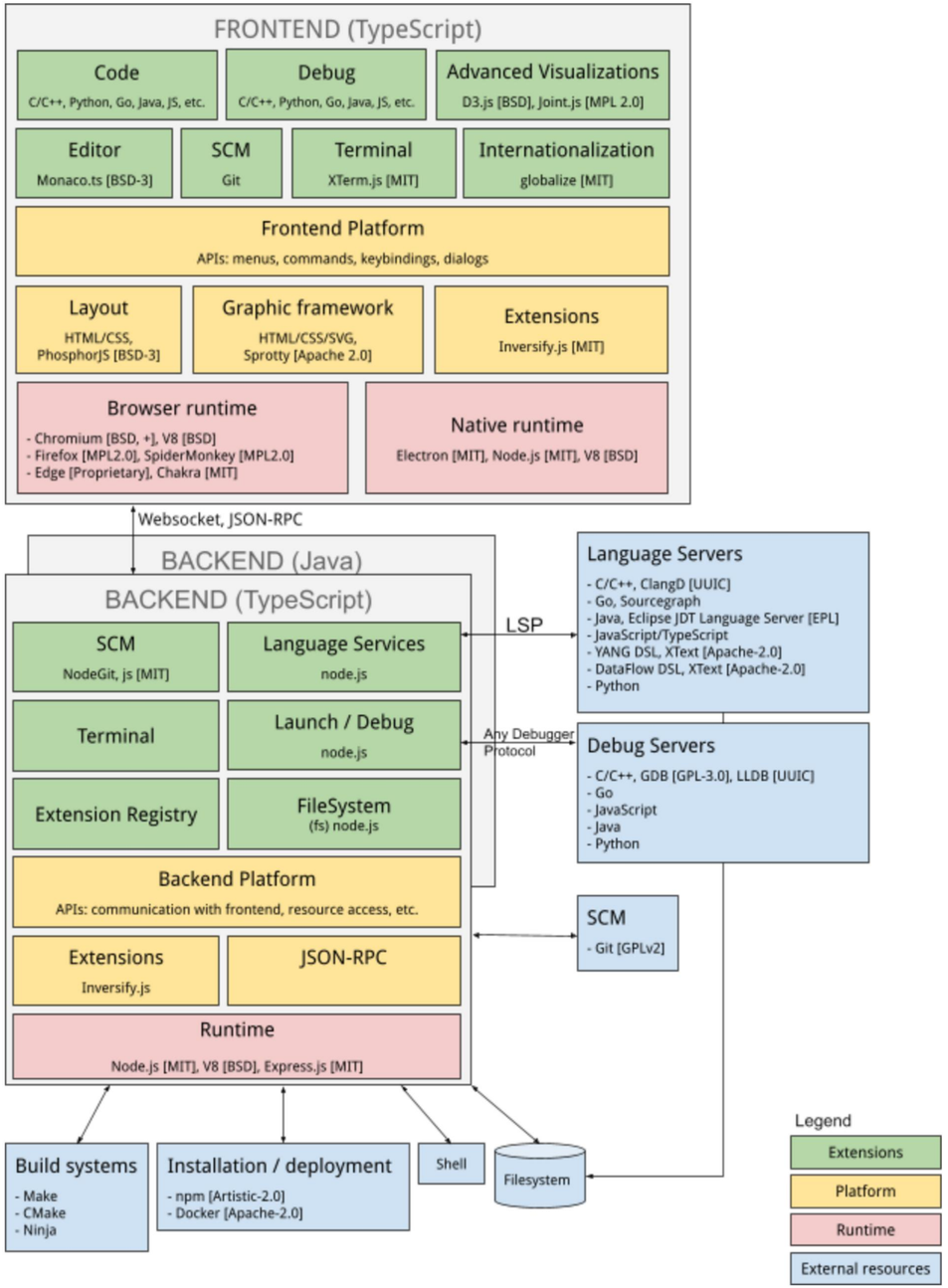
- DaoStore.tsx
- 状态管理非常复杂
- 自行实现状态机
- 共享全局数据

```
/**
 * 状态机事件：
 * transitionReady 设为可用
 * transitionOffline 下线
 * transitionOnline 上线
 * transitionError 设为错误
 */
type AppStatus =
  | 'drafted' // 初始状态
  | 'loading' // 正在初始化
  | 'ready' // 基本可用(文件树、编辑器可用)，连接成功并且已经同步完成状态
  | 'ready_offline' // 在可用状态下，网络掉线了，只有编辑器可用，其他均不可用
  | 'error'; // 发生致命问题，通知用户

type PlaygroundIDEServerStatus = 'NOT_SYNC' | 'SYNCING' | 'SYNCED' | 'ERROR';
```

异步事件编程

- SocketIO
- Channel 与服务端进行异步事件，基于状态驱动的封装与管理
- 组件的无依赖状态设计
- 离线上下线交互

```
this.socketParser = socketParser || 'default';
actions.dao.openSocketBuffer(this.socketParser === 'buffer');

let agentUserId;
try {
  agentUserId = await findAvailableAgentUserIdByUserId(userId);
} catch (e) {
  // Code here just because all the local agent IDs are being used.
}
const visibilitychange = store.dao.channel().visibilitychange;
// 1.先删除事件监听，防止多次注册事件
document.removeEventListener('visibilitychange', visibilitychange);
actions.dao.reconnectMaxTimes(this.paasOptions.reconnectMaxTimes || 10);
store.dao.channel().startChannel(
  channelPath,
  {
    ...this.paasOptions,
    agentUserId: agentUserId,
    mockSocket: this.paasOptions.socket,
  },
  {
    triggerEventForMaxTimespan: this.paasOptions.triggerEventForMaxTimespan,
    reconnectMaxTimes: this.paasOptions.reconnectMaxTimes,
    isInsertCrdt: this.paasOptions.isInsertCrdt,
    sockerParser: this.socketParser,
  },
);
```


FRONTEND (TypeScript)
Code
C/C++, Python, Go, Java, JS, etc.
Debug
C/C++, Python, Go, Java, JS, etc.
Advanced Visualizations
D3.js [BSD], Joint.js [MPL 2.0]
Editor
Monaco.ts [BSD-3]
SCM
Git
Terminal
XTerm.js [MIT]
Internationalization
globalize [MIT]
Frontend Platform
APIs: menus, commands, keybindings, dialogs
Layout
HTML/CSS,
PhosphorJS [BSD-3]
Graphic framework
HTML/CSS/SVG,
Sprotty [Apache 2.0]
Extensions
Inversify.js [MIT]
Browser runtime
- Chromium [BSD, +], V8 [BSD]
- Firefox [MPL2.0], SpiderMonkey [MPL2.0]
- Edge [Proprietary], Chakra [MIT]
Native runtime
Electron [MIT], Node.js [MIT], V8 [BSD]
Websocket, JSON-RPC
BACKEND (Java)
BACKEND (TypeScript)
SCM
NodeGit, js [MIT]
Language Services
node.js
Terminal
Launch / Debug
node.js
Extension Registry
FileSystem
(fs) node.js
Backend Platform
APIs: communication with frontend, resource access, etc.
Extensions
Inversify.js
JSON-RPC
Runtime
Node.js [MIT], V8 [BSD], Express.js [MIT]
LSP
Any Debugger
Protocol
Language Servers
- C/C++, ClangD [UUIC]
- Go, Sourcegraph
- Java, Eclipse JDT Language Server [EPL]
- JavaScript/TypeScript
- YANG DSL, XText [Apache-2.0]
- DataFlow DSL, XText [Apache-2.0]
- Python
Debug Servers
- C/C++, GDB [GPL-3.0], LLDB [UUIC]
- Go
- JavaScript
- Java
- Python
SCM
- Git [GPLv2]
Build systems
- Make
- CMake
- Ninja
Installation / deployment
- npm [Artistic-2.0]
- Docker [Apache-2.0]
Shell
Filesystem
Legend
Extensions
Platform
Runtime
External resources

Node后端职责与架构选型

后端技术栈选型

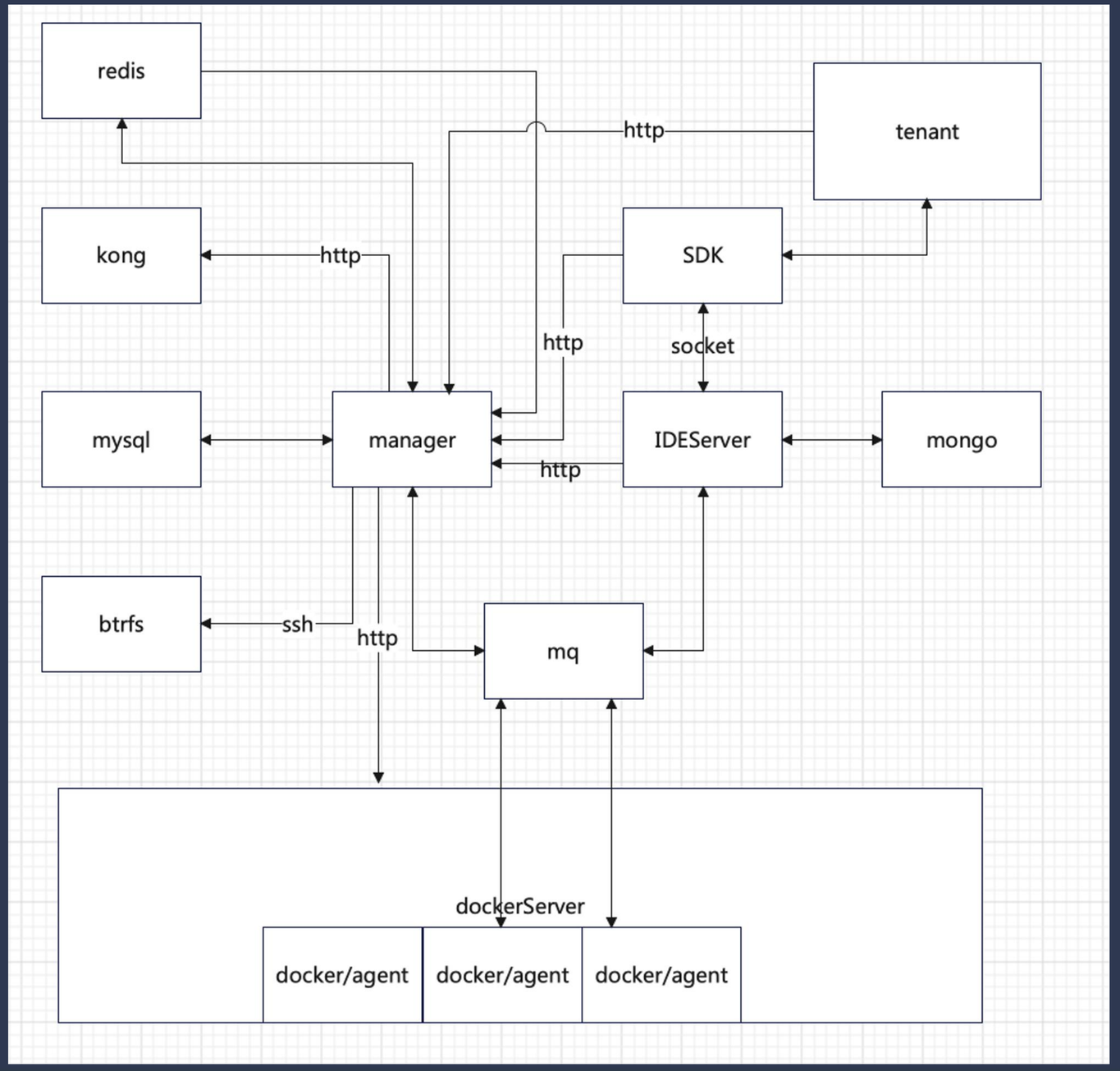
- 与 SocketIO 通信极为便捷
- 高并发支撑
- 异步事件编程支持良好
- 对 OT 协同算法支持良好，抽象性好

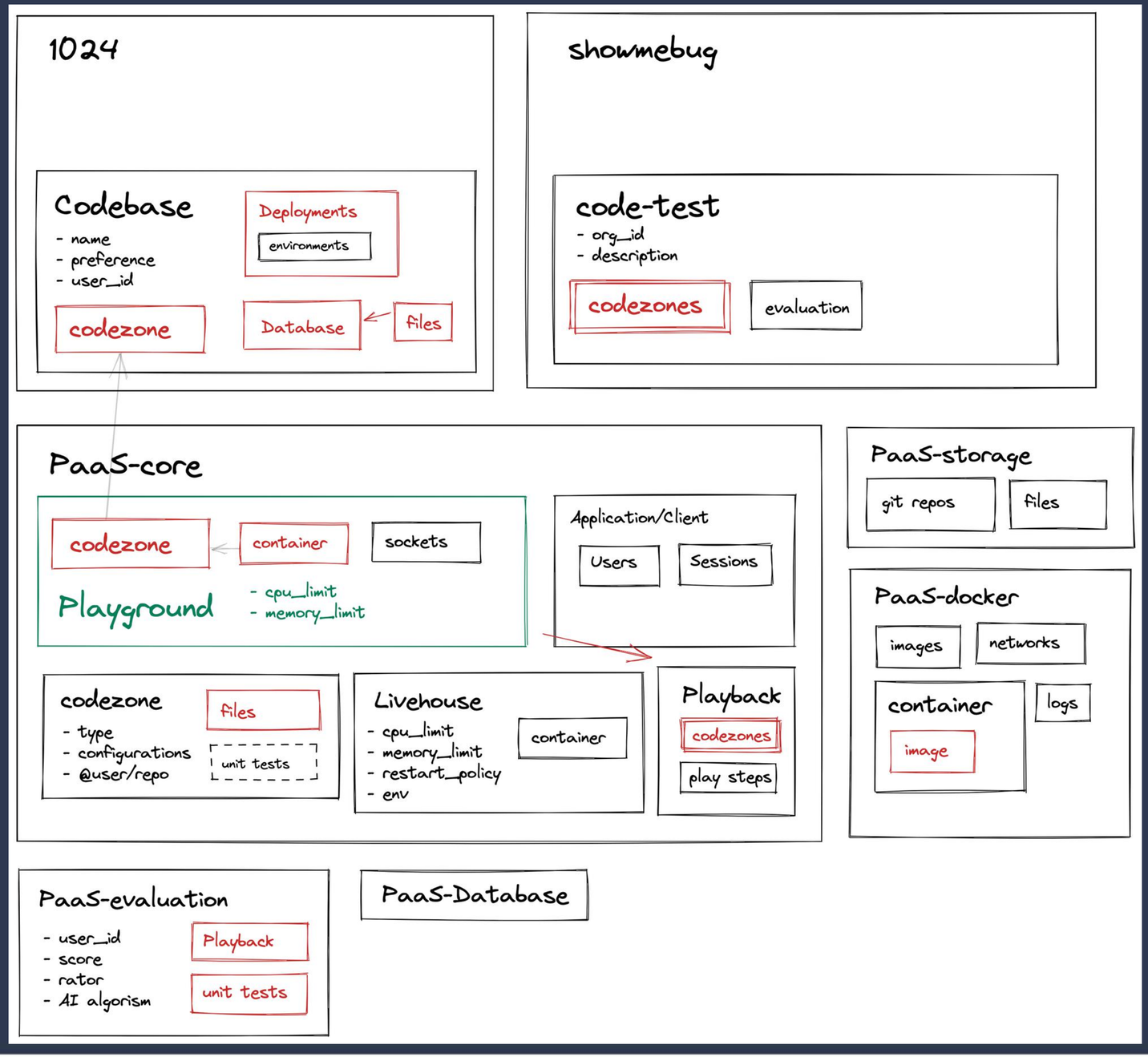
后端技术选型结论

- NestJS（充分利用 Inject 注入技术，简化复杂依赖）
- SocketIO 中间件
- 高性能支撑
- 部分复用前端状态
- Mongodb（高性能支撑，每秒2万QPS）

后端与Java中台交互选型

- MQ 消息队列（RabbitMQ NestJS中间件）
- 容器化管理机制
- 容器环境 Golang Agents 注册


redis
http
tenant
kong
http
SDK
http
socket
mysql
manager
IDEServer
mongo
http
btrfs
ssh
http
mq
dockerServer
docker/agent
docker/agent
docker/agent


1024
Codebase
- name
- preference
- user_id
Deployments
environments
codezone
Database
files
showmebug
code-test
- org_id
- description
codezones
evaluation
PaaS-core
codezone
container
sockets
Playground
- cpu_limit
- memory_limit
Application/Client
Users
Sessions
PaaS-storage
git repos
files
PaaS-docker
images
networks
container
image
logs
codezone
- type
- configurations
- @user/repo
files
unit tests
Livehouse
- cpu_limit
- memory_limit
- restart_policy
- env
container
Playback
codezones
play steps
PaaS-evaluation
- user_id
- score
- rator
- AI algorism
Playback
unit tests
PaaS-Database

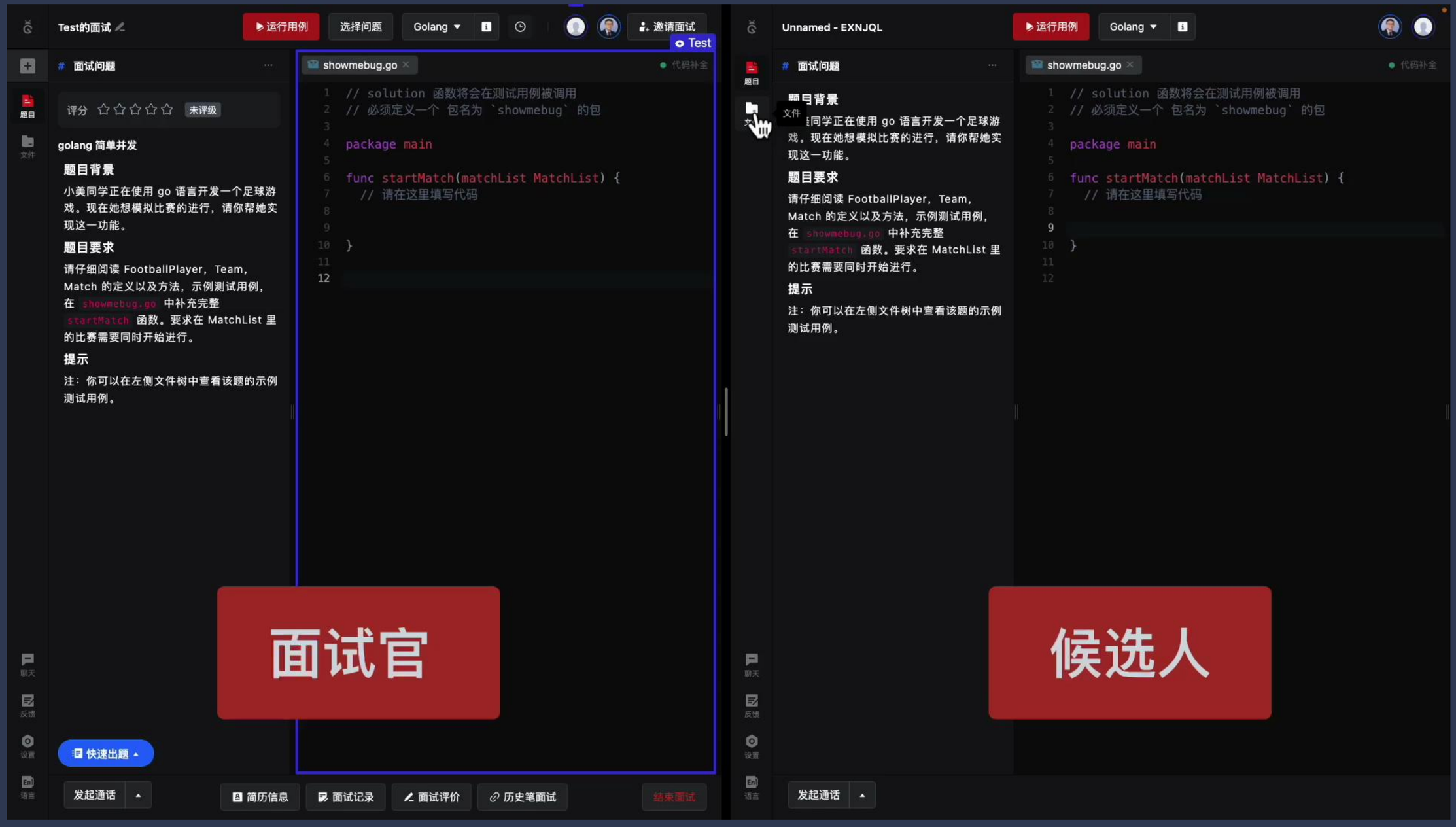
架构总结

- 单链路 1万同时在线
- 加载速度不超过1s
- 切换环境速度不超过1s
- 完美支撑业务侧灵活定制页面与接口

感兴趣可以来试试

- 前端中级以上工程师，Vue & React 有一个掌握即可构建自己的编程环境
- 你可以用 1024PaaS 实现自己的在线编程环境，通过跟我对接来体验 SDK
- 可以完整自定义各个页面结构
- 应用场景举例：在线教育、互动培训、公司内训学习


1024paas.com/demoindex
lyf
基础应用
分享协作
跟随模式
切换题目
测试用例
操作回放
编辑器配置
基础应用展示了 IDE 的基础用法，包括文件树和编辑器的使用，Console 与 Shell 的使用及它们与文件树和编辑器的联动。
选择模板
隐藏Shell
隐藏FileTree
代码处理
可以编辑答题区
显示冻结代码
显示右键菜单
添加序号
SMB
1024
markdown
markdown预览
SMB公式解析
debug
模拟初始化失败
是否模拟初始化失败
是否记录Browser数据
是否记录Browser数据
数据是否持久化
是否记录自定义文件树操作


Test的面试
运行用例
选择问题
Golang
邀请面试
Test
面试问题
showmebug.go
代码补全
评分
未评级
golang 简单并发
题目背景
小美同学正在使用 go 语言开发一个足球游戏。现在她想模拟比赛的进行，请你帮她实现这一功能。
题目要求
请仔细阅读 FootballPlayer，Team，Match 的定义以及方法，示例测试用例，在 showmebug.go 中补充完整 startMatch 函数。要求在 MatchList 里的比赛需要同时开始进行。
提示
注：你可以在左侧文件树中查看该题的示例测试用例。
// solution 函数将会在测试用例被调用
// 必须定义一个 包名为 `showmebug` 的包
package main
func startMatch(matchList MatchList) {
// 请在这里填写代码
}
面试官
快速出题
发起通话
简历信息
面试记录
面试评价
历史笔面试
结束面试
Unnamed - EXNJQL
运行用例
Golang
面试问题
showmebug.go
代码补全
文件
题目背景
同学正在使用 go 语言开发一个足球游戏。现在她想模拟比赛的进行，请你帮她实现这一功能。
题目要求
请仔细阅读 FootballPlayer，Team，Match 的定义以及方法，示例测试用例，在 showmebug.go 中补充完整 startMatch 函数。要求在 MatchList 里的比赛需要同时开始进行。
提示
注：你可以在左侧文件树中查看该题的示例测试用例。
// solution 函数将会在测试用例被调用
// 必须定义一个 包名为 `showmebug` 的包
package main
func startMatch(matchList MatchList) {
// 请在这里填写代码
}
候选人
发起通话

QA环节

- 欢迎与我交流架构、前端、创业相关的话题
- 欢迎使用 ShowMeBug 进行技术测评
- 欢迎使用 1024PaaS 开发自己的IDE应用

微信号：lyfi2003

想一想，我该如何把这些技术应用在工作实践中？

THANKS