انتخاب الگوی مناسب معماری

زیرسیستم های مورد بحث:

1. ثبت سفارش
2. انبارداری
3. مدیریت منابع
4. پرداخت
5. مدیریت اطلاعات
6. ثبت سفارش:

این قسمت به صورت یک برنامه تعاملی است که کاربران با آن ارتباط برقرار می­کنند (سفارش مورد نظر خود را انتخاب می­کنند). به نوعی می­توان این قسمت از سیستم را به سه بخش تقسیم کرد.بخش اول وظیفه دریافت اطلاعات از بخش مدیریت اطلاعات را دارد. بخش دوم وظیفه نشان دادن منو بر اساس اطلاعات دریافتی را بر عهده دارد. بخش سوم قابلیت انتخاب از منو را به کاربر می­دهد.

بر اساس این مسائل الگوی معماری MVC(Model-View-Controller) مناسب است اما برای پردازش سفارشهای چند کاربر به صورت همزمان لازم است که از مدل PAC(Presentation-Abstraction-Control) استفاده کنیم.

مزایا:

• مناسب ترین روش برای پردازش های توزیع شده در یک شبکه با تعداد کاربران زیاد

• سهولت در امر پیاده سازی

• نسبت دهی مستقیم رابط کاربر با منابع تامین داده ها

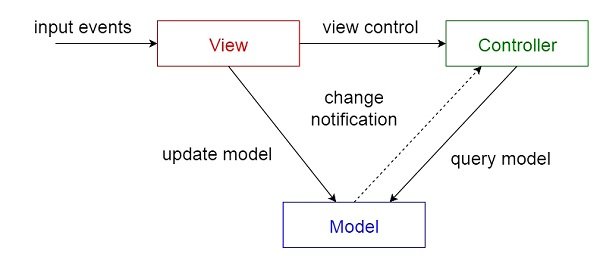
معایب:

• کاهش کارائی برنامه همزمان با افزایش تعداد کاربران همزمان

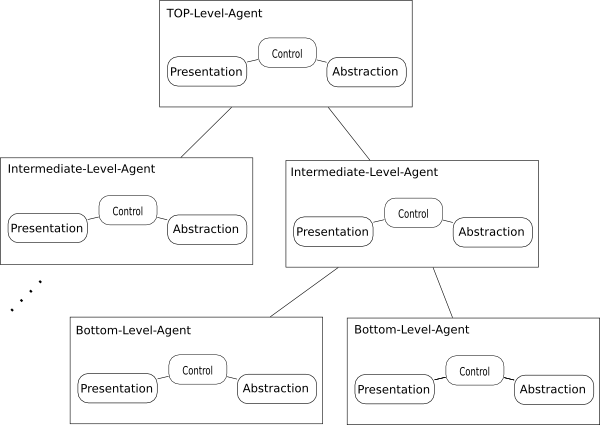
MVC:

این الگو که اغلب موارد با واژه MVC از آن نام‌برده می‌شود، یک برنامه تعاملی را به سه بخش زیر تقسیم می‌کند.

1. Model: که شامل قابلیت‌های اصلی برنامه و داده‌ها است.
2. View: وظیفه این بخش نشان دادن اطلاعات به کاربر است. در برخی از برنامه‌ها ویو نقشی بیش از نشان دادن اطلاعات بر عهده دارد.
3. Controller: این بخش مدیریت داده‌های ورودی را که از سوی کاربران وارد می‌شود، برعهده دارد. همچنین وظیفه برقراری ارتباط میان model و view نیز بر عهده مولفه کنترلر است.



PAC: مانند MVC با این تفاوت که هر کدام از برنامه‌ها(agent) از طریق بخش control با یکدیگر تعامل دارند؛ تفاوت دیگر اینکه presentation و abstraction کپسوله هستند که امکان multithreading را ممکن می‌سازد.



1. انبارداری:

این قسمت از سیستم مسئول کنترل موجودی کالاها، تاریخ­های انقضا، درخواست خرید به مدیریت منابع و غیره است. به دلیل سرویس های مختلفی که این سیستم ارائه می­دهد، الگوی معماری Microservice مناسب است.

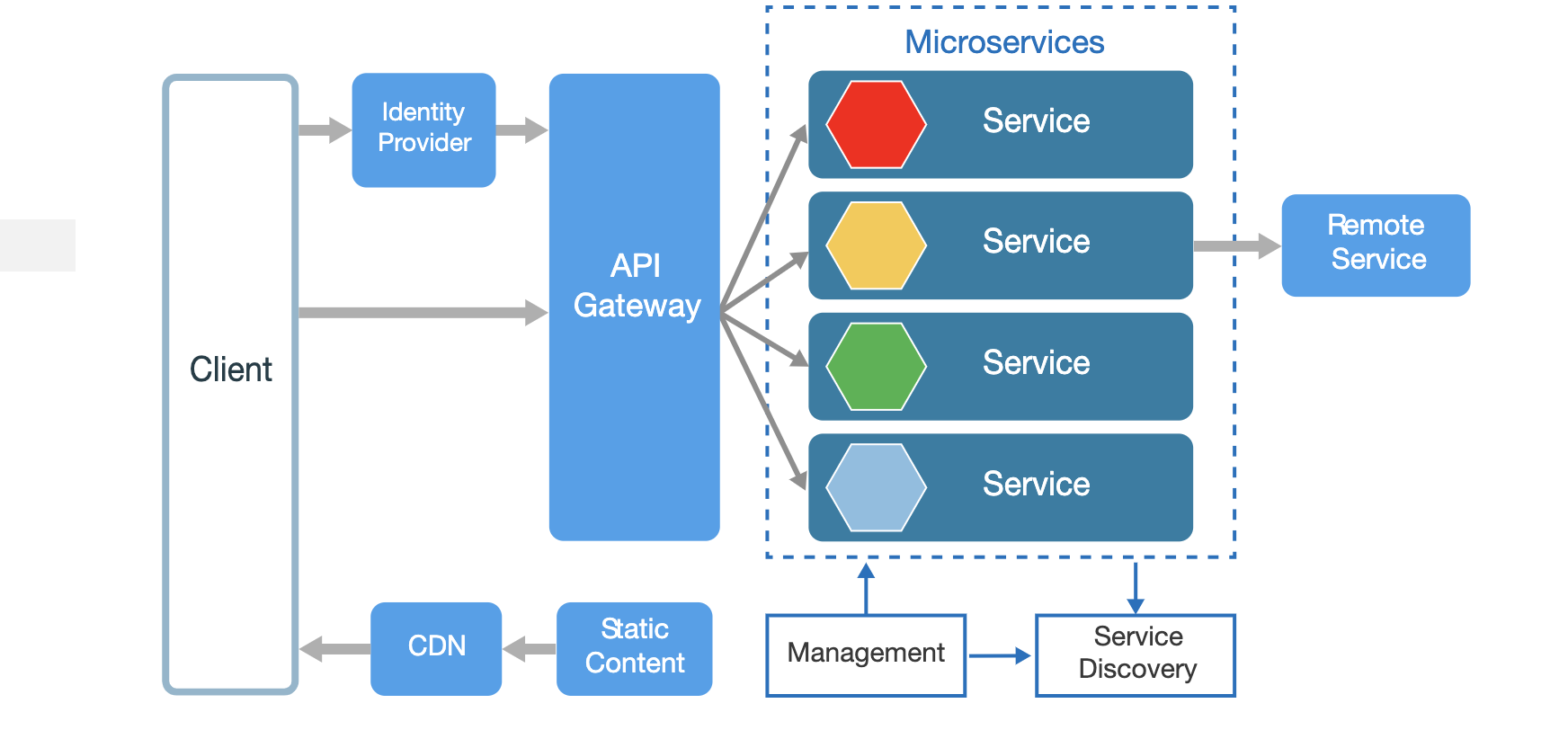
مزایا:

* به دلیل ایزوله بودن سرویسها درصورت اختلال در یک سرویس، برای بقیه سرویسها اشکالی ایجاد نمی­شود
* انعطاف پذیری هنگام افزایش یا کاهش سرویسها
* راحتی فهم کاربرد سیستم
* Agility
* Scalability

معایب:

* تداخل بین سرویسها
* دشواری تست
* استفاده بیشتر از منابع

Microservice: معماری مایکروسرویس یک رویکرد چابک و ماژولار به توسعه نرمافزار است که برخلاف نرمافزارهای یکپارچه که همه مولفه­ها و قابلیتهای سیستم بایکدیگر آمیخته شده­اند. به عبارتی مبتنی بر مجموعه­ای از سرویسهای کوچکتر و مستقل از هم با ارتباط سست است.



1. مدیریت منابع:

در این قسمت از سیستم باید درخواست خرید کالاهای مورد نیاز از سمت سیستم به سمت شرکت‌های تامین‌کننده فرستاده شود و تاییدیه از سمت آن‌ها دریافت شود. پس معماری مناسب این قسمت، client-server است.

مزایا:

* مناسب ترین روش برای پردازش­های توزیع شده
* سهولت در امر پیاده سازی

معایب:

* فشار محاسباتی در قسمت server
* مشکلات مربوط به برقراری ارتباط با سرور

1. پرداخت:

در این قسمت از سیستم باید درخواست پرداخت از سمت کاربر به سمت ارائه دهنده سرویس فرستاده شود تا عملیات پرداخت از سمت سرور انجام بگیرد. پس معماری مناسب این قسمت، client-server است.

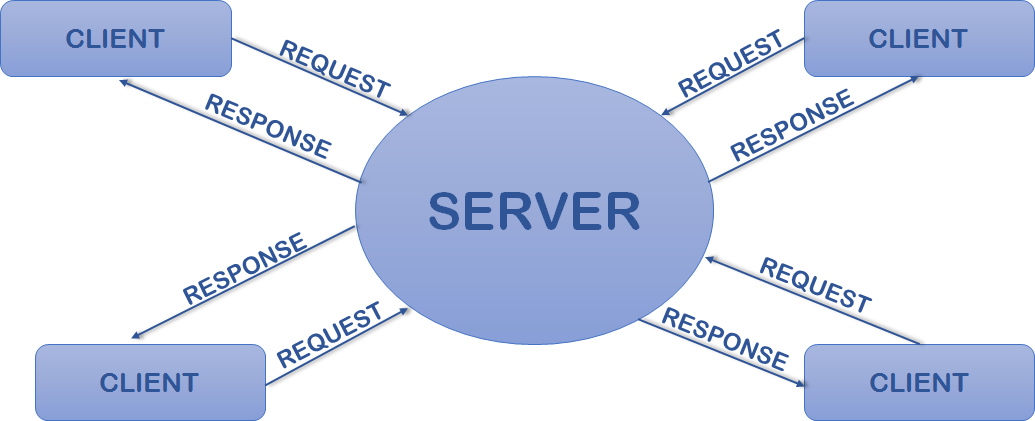
مزایا:

* مناسب ترین روش برای پردازش­های توزیع شده
* سهولت در امر پیاده سازی

معایب:

* فشار محاسباتی در قسمت server
* مشکلات مربوط به برقراری ارتباط با سرور

Client-Server: در این مدل از یک سرویس دهنده و یک سرویس گیرنده در شبکه استفاده می گردد و منطق برنامه بین دو محل فیزیکی توزیع می گردد.



1. مدیریت اطلاعات:

در این قسمت از سیستم باید اطلاعات بر اساس نوع درخواست کننده ارائه شود. که بر این اساس مدل Broker مناسب این قسمت است.

مزایا:

معایب:

Broker: این الگو برای ساخت سامانه‌های توزیع‌شده با مولفه‌های جدا از هم (decoupled) کاربرد دارد.  یک مولفه کارگزار وظیفه هماهنگ کردن ارتباط میان سایر مولفه‌ها را عهده‌دار است. در این الگوی معماری سرورها می‌توانند قابلیت‌ها، توانایی‌ها، سرویس‌ها و مشخصات خود را به کارگزار واگذار کنند. در این حالت زمانی که کلاینت‌ها سرویسی را از یک کارگزار درخواست می‌کنند، کارگزار کلاینت‌ را به یک سرویس مناسب هدایت می‌کند.

