یکپارچه سازی کاربرد های سازمانی

Created by Hamed Mirzaei

Github: https://github.com/HamedMirzaeiOfficial

Telegram: https://t.me/HamedMirzaeiOfficial

معماری سازمانی: رویکردی جامع و یکپارچه است که جنبه ها و عناصر مختلف یک سازمان را با نگاه مهندسی تفکیک و تحلیل میکند و شامل مجموعه مستندات، مدل ها، استانداردها و اقدامات اجرایی برای تحول از وضعیت موجود به وضعیت مطلوب با محوریت فناوری اطلاعات است و در قالب یک طرح اجرا شده و سپس به صورت مداوم توسعه و بروزرسانی میشود.

معماری سازمانی چه چیز هایی را در نظر میگیرد؟ تمام جنبه های سازمان نظیر کاربران، موقعیت جغرافیایی سیستم ها، نحوه توزیع آنها، فرایند های حرفه، انگیزه کار ها، راهبرد ها، ماموریت های سازمان و ...

تفاوت معماری سازمانی و معماری اطلاعات: معماری سازمانی نوعی سازماندهی مجدد را در کل سازمان از منظر سیستم های اطلاعاتی در جهت بهبود فرایند های کاری سازمان از طریق به کارگیری فناوری اطلاعات شکل میدهد. در حالیکه تمرکز معماری اطلاعات روی اطلاعات جاری در سازمان است.

فرایند معماری شامل سه فاز اصلی: ۱- برنامه ریزی راهبردی فناوری اطلاعات ۲- برنامه ریزی معماری سازمانی ۳- اجرای معماری سازمانی

مزایای معماری سازمانی در مقیاس کشوری(دولت الکترونیک):

۱- نگاه جامع کشوری به راهکار ها، استاندارد ها و فناوری ها

۲- هماهنگ سازی معماری سازمان ها با معماری دولت الکترونیک

۳- تحقق تعامل پذیری بین سازمانی

۴- کاهش هزینه ها و جلوگیری از دوباره کاری در صنعت فاوا

۵- تحقق ارائه خدمات یکپارچه از پنجره واحد به شهروندان

۶- زمینه سازی برای پایش خدمات الکترونیک دستگاه ها و بهبود مستمر شاخص ها

مزایای معماری سازمانی برای دستگاه های دولتی و شرکت های خصوصی:

۱- هم راستا سازی اهداف و فعالیت های فاوا با ماموریت و اهداف سازمانی(کارامدی سازمان)، یکپارچگی خدمات، سیستم ها و بانک های اطلاعات سازمانی(یکپارچگی)

۲- نگاه جامع در انتخاب راهکار های فاوا مناسب و متناسب (ERP, BPMS, Portal, ITIL, ISMS)

۳- مدیریت پذیری و کاهش هزینه های فناوری اطلاعات

۴- ارائه خدمات الکترونیکی کارامد و یکپارچه سازمانی به شهروندان

هدف فرایند معماری سازمانی: ایجاد و اجرای معماری و ارائه خروجی های معماری در سازمان

مراحل فرایند معماری سازمانی:

۱- برنامه ریزی راهبردی فناوری اطلاعات

۲- برنامه ریزی معماری سازمانی

۳- اجرای معماری سازمانی

اولین چارچوب معماری سازمانی: چارچوب معماری zachman در سال ۱۹۸۷

چارچوب ها و مدل های مرجع معماری سازمانی عمومی(همه منظوره): این چارچوب ها برای دولت یا صنعت خاصی تولید نشده اند و به صورت عمومی برای سازمان ها و مقاصد مختلفی قابل استفاده هستند. البته به دلیل عمومی بودن نیاز به سفارشی سازی در صنعت یا کاربرد خاص هستند.

معروف ترین چارچوب ها و مدل های مرجع معماری سازمانی عمومی(همه منظوره):

۱- The Zackman Framwework

۲- TOGAF

۳- GEAF

۴- OEAF

یکپارچه سازی کاربرد های سازمانی(Enterprise Application Integration: EAI): مجموعه ای است از فرایند ها، استانداردها، نرم افزار ها و سخت افزار ها که در راستای یکپارچه سازی دو یا چند سیستم کاربردی سازمان(نرم افزار کاربردی) عمل نموده و برای این سیستم ها شرایطی را فراهم میسازد تا بتوانند در قالب یک سیستم واحد عمل کنند.

فواید یکپارچه سازی سازمانی:

۱- به سیستم ها و نرم افزار ها کمک میکنه تا بتوانند به صورت همزمان و لحظه ای به اطلاعات یکدیگر دسترسی داشته باشند.

۲- با ساده سازی فرایند های کسب و کار، راندمان و کارایی سازمان را افزایش میدهد.

۳- یکپارچه سازی اطلاعات را فراهم میکند.

۴- توسعه و نگهداری سیستم ها را آسان میسازد.

۵- ارتباطات با مشتری را بهبود میبخشد.

۶- ارتباط با زنجیره تامین (supply chain) را بهبود میبخشد.

۷- نرم افزار های کاربردی قدیمی رو همچنان فعال و زنده نگه میدارد.

۸- فرایند های کسب و کار را بهبود میبخشد و در نتیجه مدت زمان مورد نیاز برای عرضه محصول به بازار (time to marke) را کاهش میدهد.

۹- تا حدودی از تغییرات در سطح فرایندی یا سازمانی پشتیبانی میکند.

۱۰- نرم افزار های کاربردی رو استانداردسازی میکند.

۱۱- تکنولوژی های واکنشی(responsive technology) را برای نیاز های متغیر کسب و کار به خدمت میگیرد.

انواع یکپارچه سازی کاربرد های سازمانی:

۱- سطح داده(Data Level): این سطح از یکپارچه سازی شامل مجموعه ای از تکنیک ها، فرایند ها و تکنولوژی هاست که انتقال داده میان منابع داده را امکان پذیر میکند. مزیت اصلی آن، عدم نیاز به تغیر source code است که باعث کاهش هزینه های توسعه میشود.

۲- سطح رابط برنامه کاربردی(Application Interface Level): در این سطح با استفاده از رابط های نرم افزاری که هرکدام شامل توابع و خصوصیات مشخصی هستند، امکان اشتراک گذاری منطق تجاری و اطلاعات نرم افزار ها وجود خواهد داشت.

۳- سطح متد(Method Level): در این سطح منطق تجاری نرم افزار ها به اشتراک گذاشته میشه. یک متد میتونه توسط تعداد زیادی از نرم افزار های کاربردی قابل دسترس باشه و همچنین نرم افزار ها میتونن به متد های یکدیگر دسترسی داشته باشن.

۴- سطح رابط کاربری(User Interface Level): در این روش معماران و توسعه دهندگان میتونن از رابط های کاربری به عنوان یک نقطه اشتراک جهت یکپارچه سازی استفاده کنند(استفاده از رابط کاربری مشترک برای نرم افزار ها)

دسته بندی یکپارچه سازی کاربرد های سازمانی(EAI) بر اساس طراحی ساختار:

۱- توپولوژی Point To Point: به عنوان یک روش یکپارچه سازی قدیمی شناخته میشه. به نرم افزار ها این امکان رو میده که با استفاده از یه لوله(pipe) به هم متصل شن. هر نرم افزار کاربردی با استفاده از یک پیام یا یک رویه فراخوانی(call procedure) میتواند با نرم افزار مقابل خود ارتباط برقرار کند. به ازای هر جفت از نرم افزار های مرتبط، یک اتصال دهنده(connector) به منظور برقراری ارتباط، ساخته و پیاده سازی میشود. این اتصال دهنده وظیفه ی تبدیل و یکپارچه سازی داده برای هر جفت معین از نرم افزار ها را برعهده دارد. البته امکان اتصال بیش از دو نرم افزار کاربردی هم وجود داره، اما این کار پیچیدگی بسیاری را به همراه خواهد داشت(در صورت زیاد بودن تعداد نرم افزار ها)

۲- توپولوژی Hub Spoke یا Broker: از یه واسط متمرکز(Hub) و تعدادی تطبیق دهنده یا آداپتور(Spoke) تشکیل شده است. در واقع spoke اتصال دهنده ای است که نرم افزار کاربردی را به hub متصل میکند و داده ها را به منظور برقراری ارتباط میان نرم افزار کاربردی و hub ترجمه مینماید. به این صورت که پیام ها تبدیل شده و ترجمه شده و به سمت مقصد(نرم افزار به hub یا برعکس) هدایت میگردند.

۳- توپولوژی Bus: طبیعت متمرکز مدل hub spoke یا broker تنها نقطه ی ضعف این مدله، چونکه اگر یک مولفه(component) دچار مشکل بشه، باعث ایجاد نقص در تمام شبکه میشه. مدل bus به عنوان راه حلی برای مشکلات مدل broker ایجاد شده. این مدل هم از یک مولفه ی متمرکز استفاده میکنه با این تفاوت که مابقی وظایف را میان سایر مولفه ها تقسیم و توزیع میکنه. این مولفه ها میتونن در نقاط مختلفی از شبکه، گروه بندی و میزبانی بشن. از قابلیت های دیگه ی این مدل میتونیم به پردازش تراکنش امنیتی(security transaction processing) و قابلیت رفع خطا(error handling) اشاره کرد. مدل bus یک راهکار مختصر با الگویی مستحکمه که میتونه با کمترین حجم کدنویسی و بدون اعمال تغییر در نرم افزار کاربردی، طراحی و مورد استفاده قرار بگیره. امروزه این مدل با نام ESB(Enterprice Service Bus) شناخته میشه.

برنامه ریزی منابع سازمان(ERP:Enterprise Resource Planning): طیف وسیعی از فعالیت های مختلف رو شامل میشه که هدف آن، گرداوری تمام داده ها و فرایند های یک سازمان در یک سیستم واحد و در نهایت بهبود عملکرد سازمان میباشد. در واقع ERP سامانه ی یکپارچه ای است که دارای اهداف، اجزا و محدوده مشخص و معینی میباشد.

یک راه حل نرم افزاریه که تمام فعالیت های واحد های مختلف سازمان را به طور یکپارچه در یک سیستم نرم افزاری واحد تعریف و ایجاد میکند.

مهمترین ویژگی یک ERP فرایند گرا بودن اونه.

مزایای ERP:

۱- کاهش هزینه های حمل موجودی

۲- کاهش هزینه های سفارش

۳- کاهش هزینه های تولید

۴- کاهش هزینه های نگهداری سوابق

۵- کاهش هزینه های حمل و نقل

۶- کاهش سرمایه گذاری در تجهیزات

۷- فرایند های تولید انعطاف پذیر

۸- بهبود کارایی که به سود دهی بیشتر یا افزایش سهم بازار منجر میشه.

۹- افزایش شفافیت فرایند برای مشتری

۱۰- افزایش رضایت مشتری

معایب ERP:

۱- نصب و نگهداری این سیستم ها بسیار گران است.

۲- استفاده از بعضی از این سیستم ها دشوار است.

۳- برای به اشتراک گذاشتن برخی از اطلاعات حساس که برای یک فرایند ضروری است، با مقاومت فراد مواجه میشوید.

مدیریت فرایند های کسب و کار(Business Process Management : BPM): عبارت اند از تحلیل فرایند های کسب و کار با استفاده از مدل ها و سناریوهای متفاوت به منظور درک شیوه ی انجام فعالیت ها توسط سازمان های بزرگ. این امر موجب ایجاد بینشی جدید برای ارائه توصیه و آزمون بهینه سازی ها و بهبود های سازمانی شده و تکرار مجدد فرایند برای بهبود مداوم و بهینه سازی فرایند را باعث میشود.

مزایای BPM :

۱- توسعه و تعالی سازمانی

۲- ایجاد مزیت رقابتی و جریان ارزش پایدار

۳- جلب رضایت مشتریان

۴- چابکی سازمان و فرایند های سازمانی

۵- یکپارچه سازی سازمان و فرایند های سازمانی

لایه ی ip: هدایت و مسیر یابی بسته های اطلاعاتی از یک ماشین میزبان به ماشین دیگر.

لایه ی انتقال: فراهم آوردن خدمات سازماندهی شده مبتنی بر اصول سیستم عامل، برای برنامه های کاربردی لایه بالاتر. جبران کاستی های لایه ip.

پروتکل های لایه انتقال:

1- UDP: User Datagram Protocol

2- TCP: Transmission Control Protocol

کاستی های لایه ی ip و راهکار های پروتکل TCP:

۱- عدم تضمین در آماده بودن ماشین مقصد جهت دریافت بسته : برقراری یک ارتباط و اقدام به هماهنگی بین مبدا و مقصد قبل از ارسال هرگونه داده

۲- عدم تضمین در ترتیب رسیدن بسته های متوالی و همچنین صحت داده ها: قراردادن شماره ترتیب برای داده ها – تنظیم کد ۱۶ بیتی کشف خطا در مبدا و بررسی مجدد آن در مقصد جهت اطمینان از صحت داده ها.

۳- عدم تمایز در دریافت بسته های تکراری در مقصد(duplication problem): قراردادن شماره ترتیب در بسته ارسالی

۴- عدم تنظیم سرعت ارسال و تحویل بسته ها: استفاده از الگوریتم پویا جهت تنظیم مجموعه زمانسنج ها

۵- عدم توزیع بسته ها بین پروسه های مختلف اجرا شده بر روی یک ماشین واحد: قراردادن آدرس پورت پروسه فرستنده و گیرنده در سرایند بسته ارسالی

آدرس پورت: شماره شناسایی مشخص کننده ی هر پروسه برای برقراری یک ارتباط با پروسه ی دیگر بر روی شبکه.

TCP و UDP از ۱۶ بیت برای پورت ها استفاده میکنند بنابراین 65535 پورت ممکن داریم.

۱-۱۰۲۳: برای پورت های سیستم

۱۰۲۴-۴۹۱۵۱: برای پورت های کاربر

۴۹۱۵۲–۶۵۵۳۵: پورت های دینامیک

شماره پورتهای استاندارد

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

آدرس سوکت: زوج ادرس ip و آدرس پورت مشخص کننده یک پروسه یکتا و واحد بر روی یک ماشین در دنیا.

سوکت: یک مفهوم انتزاعی از تعریف ارتباط در سطح برنامه نویسی- اعلام آمادگی جهت مبادله داده ها توسط برنامه نویس به سیستم عامل بدون درگیر شدن با جزئیات پروتکل TCP و UDP و تقاضای ایجاد فضا و منابع مورد نیاز جهت برقراری یک ارتباط از سیستم عامل.

روال برقراری ارتباط بین دو برنامه از راه دور:

الف: درخواست برقراری ارتباط با کامپیوتری خاص با ip مشخص و برنامه ای روی آن کامپیوتر با آدرس پورت مشخص(درخواست فراخوانی تابع سیستمی socket)

ب) مبادله داده ها با توابع send و recv در صورت برقراری ارتباط

ج) اتمام ارتباط با فراخوانی تابع close

انواع سوکت ها:

۱- سوکت های نوع استریم = سوکت های اتصال گرا connection oriented – مبتنی بر پروتکل TCP – لزوم برقراری یک اتصال قبل از مبادله ی داده ها به روش hande shake سه مرحله ای.

کاربرد در: پروتکل های انتقال فایلFTP، انتقال صفحات ابرمتنHTTP، انتقال نامه های الکترونیکی SMTP

۲- سوکت های نوع دیتاگرام = سوکت های بدون اتصال connectionless – مبتنی بر پروتکل UDP – مبادله داده بدون نیاز به برقراری هیچ ارتباط یا اتصالی و عدم تضمین در رسیدن داده ها، صحت داده ها و ترتیب داده ها

کاربرد در: انتقال صدا و تصویر یا سیستم DNS

مشتری(client): پروسه ای نیازمند اطلاعاته.

سرویس دهنده(server): پروسه ای برای اشتراک گذاری اطلاعات و تحویل به مشتریه.

برنامه سمت سرور(server side): برنامه ای که روی ماشین سرویس دهنده نصب میشه و منتظر تقاضای برقراری ارتباطه و بعد از پردازشش، پاسخ مناسبو ارسال میکنه. پس در حالت کلی سرویس دهنده شروع کننده ی یک ارتباط نیست.

برنامه سمت مشتری: برنامه های سمت مشتری بنا به نیاز اقدام به درخواست اطلاعات میکنند. تعداد مشتری ها روی یک یا چند ماشین میتواند متعدد باشد ولی تعداد سرویس دهنده ها معمولا یکی است.(مگر در سیستم های توزیع شده)

مثال سوکت زدن با استفاده از UDP:

client.py:

from socket import * // اضافه کردن ماژول به برنامه

server_name = ‘127.0.0.1’ // مشخص کردن نام سرور

server_port = 12000 // مشخص کردن پورت سرور

client_socket = socket(AF_INET, SOCK_DGRAM) // ایجاد سوکت udp

message = input(“Enter a lowercase sentence: ”) // گرفتن یک متن از کاربر

client_socket.sendto(message, (server_name, server_port)) // ارسال پیام دریافت شده از کاربر به آدرس 127.0.0.1 و پورت 12000

modified_message, server_address = clinet_socket.recvfrom(2048) // دریافت پیام تغیر داده شده در سرور که تبدیل به حالت اپرکیس یا حروف بزرگ شده. و همچنین دریافت آدرس سرور

print(modified_message) // نمایش پیام تغییر داده شده

cilent_socket.close()// بستن سوکت

server.py:

from socket import * // اضافه کردن همه ی موارد موجود در ماژول socket

server_port = 12000 // تعیین پورت سرور

server_socket = socket(AF_INET, SOCK_DGRAM) // ایجاد سوکت udp

server_socket.bind((‘’, server_port)) // گوش دادن سرور به پورت 12000

```
print("the server is ready to receive something") // سرور آماده ی دریافت چیزیست
while 1: // حلقه ی بینهایت
        message, client_address = server_socket.recvfrom(2048)  // دریافت پیام و آدرس کلاینت ارسال کننده ی پیام
        modified_message = message.upper() // تبدیل حروف پیام به حروف بزرگ الفبا
        server_socket.sentto(modified_message, client_address) // ارسال پیام تغییر داده شده به سمت کلاینت ارسال کننده ی آن
```

مثال سوکت زدن با استفاده از TCP :

client.py:

```
from socket import * // افزودن تمام موارد موجود در ماژول سوکت
sever_name = '127.0.0.1' // تعیین ادرس سرور برای کلاینت
server_port = 12000 // تعیین ادرس پورت برای کلاینت
client_socket = socket(AF_INET, SOCK_STREAM) // ایجاد سوکت برای کلاینت به روش tcp
client_socket.connect((server_name, server_port)) // وصل شدن و هند شیک دادن
sentence = input("Enter lowercase sentence: ") // گرفتن جمله از کاربر
client_socket.send(sentence) // ارسال جمله به سرور
modified_sentence = client_socket,recv(1024)  // دریافت پیام تغییر داده شده از سرور

print("from server: ", modified_sentence) // چاپ پیام
client_socket.close() // بستن سوکت
```

server.py:

```
from socket import * // افزودن تمام موارد موجود در ماژول سوکت
server_name = '127.0.0.1' // تعیین آدرس سرور
server_port = 12000 // تعیین پورت سرور
server_socket = socket(AF_INET, SOCK_STREAM)  // ایجاد سوکت برای سرور برای هندشیک دادن اولیه به روش tcp
server_socket.bind(('', server_port)) // ادامه فرایند هند شیک اولیه و گوش دادن به پورت
server_socket.listen(1) // ادامه فرایند هند شیک اولیه و گوش دادن به پورت برای دریافت درخواست از طرف کلاینت
print("The server is ready to receive") // چاپ اینکه سرور اماده دریافت درخواسته
while 1: // حلقه بینهایت
        connection_socket, address = server_socket.accept() // پذیرش درخواست کلاینت و ذخیرع ادرس و سوکت ایجاد شده برای ارسال پیام
        sentence = connection_socket.recv(1024) // دریافت جمله ارسالی از طرف کلاینت
        capitalized_sentence = sentence.upper() // تبدیل به حروف بزرگ
        connection_socket.send(capitalized_sentence) // ارسال آن برای کلاینت
        connection.close() // بستن سوکت ارسال پیام .
        //ادامه حلقه و تکرار این مراحل ارسال پیام جدید
```

نکته:

AF_UNIX : برای سیستم عامل یونیکس

AF_INET : برای تعیین رنج ایپی ورژن ۴

AF_INET6 : برای تعیین رنج ایپی ورژن ۶

SOCK_STREAM : برای ارتباط TCP

SOCK_DGRAM : برای ارتباط UDP

setdefaulttimeout(seconds) like ==> setdefaulttimeout(20) : تعیین تایم اوت پیشفرض برای همه سوکت ها

میتونیم یه سوکت از قبل تعریف شده داشته باشیم و براش تایم اوت ست کنیم:

s.settimeout(20)