سند معماری نرم­افزار

سیستم خرید اینترنتی

پیام الماسیان

علیرضا حریری

|  |  |  |  |
| --- | --- | --- | --- |
| تاریخ | نسخه | توضیحات | نویسنده |
|  | 1.0 | نگارش اولیه­ی سند | علیرضا حریری |
|  |  |  |  |

فهرست

[1- مقدمه](#_Toc437505378) 4

[1-1- هدف](#_Toc437505379) 4

[1-2- حیطه](#_Toc437505380) 4

[1-3- توصیفات، کلمات اختصاری و مخفف­ها](#_Toc437505381) 4

[1-4- منابع](#_Toc437505382) 4

[1-5- دید کلی](#_Toc437505383) 4

[2- نمایش معماری](#_Toc437505384) 4

[3- اهداف و محدودیت­های معماری](#_Toc437505385) 5

[4- نمای Use-Case](#_Toc437505386) 6

[5- نمای منطقی](#_Toc437505387) 8

[5-1- بسته­های معماری](#_Toc437505388) 9

[6- نمای فرآیند](#_Toc437505389) 9

[7- نمای فیزیکی](#_Toc437505390) 9

[8- نمای پیاده­سازی](#_Toc437505391) 10

[9- نمای داده](#_Toc437505392) 11

[10- کارایی و اندازه](#_Toc437505393) 12

[11- کیفیت](#_Toc437505394) 13

# 1- مقدمه

در این مستند سعی شده است که دیدی بالا از سیستم خرید اینترنتی داده شود. به این معنی که ساختار و معماری کلی سیستم در قسمت­های مختلف نیز componentها به چه صورت هست. همچنین بخش­های مختلف چگونه با یکدیگر در تعامل هستند و ساختار به صورت کلی به چه صورت است.

## 1-1- هدف

هدف از نگارش و تولید چنین سندی پیدا کردن دیدی کلی از سیستم در حال ساخت می­باشد. به این منظور که ساختار کلی سیستم و نیز تعاملات بخش­های متفاوت را در سیستم بتوان دریافت کرد. به وسیله­ی این مستند می­توان سیستم نهایی را به گونه­ای صحیح تصور نمود و نیز راه رسیدن به آن و معیارهایی را که برای ارزیابی محصول نهایی است، دریافت.

## 1-2- حیطه

این مستند بر روی شیوه­ی ساخت و نیز طراحی­های بخش­های مختلف سیستم (چه نرم­افزاری و چه سخت­افزاری) تأثیرگذار خواهد بود. این سیستم همچنین این امکان را به افراد دخیل در پروژه می­دهد که بر چه اساسی سیستم خرید اینترنتی تولید شده است و در نهایت با چه معیارهایی باید ارزیابی صورت پذیرد.

## 1-3- توصیفات، کلمات اختصاری و مخفف­ها

در ادامه­ی مستند از اصطلاح MVC که یک الگوی معماری است که بیشتر برای برنامه­های تحت وب استفاده می­گردد، استفاده می­شود.

## 1-4- منابع

مستند مدل داده (ویرایش 1)

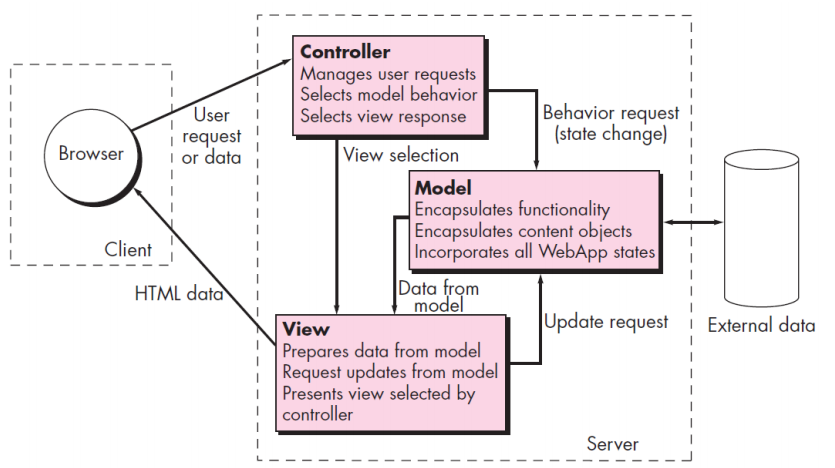
## 1-5- دید کلی

این مستند معماری بر اساس چندین الگوی طراحی معماری ساختار سیستم را مورد بررسی قرار می­دهد و در ادامه، دلیل انتخاب هر یک بر اساس معیارهای ارزیابی که در انتها آمده است، توضیح داده می­شود.

# 2- نمایش معماری

بر اساس سیستم خرید اینترنتی باید این امکان برای کاربران وجود داشته باشد که بتوانند به راحتی خرید خود را انجام دهند. از سوی دیگر ماهیت سیستم خرید اینترنتی مبتنی بر وب است. معماری که به عنوان معماری کلی برای طراحی سیستم مدنظر قرار گرفته است، معماری MVC می­باشد. به این صورت که کل سیستم شامل 3 بخش کلی model، view و controller است. از سوی دیگر همان طور که بیان شد، کل سیستم مبتنی بر وب است، بنابراین سیستم باید بر پایه­ی معماری client/server نیز قرار گیرد. به این ترتیب کاربران تنها به یک browser برای اتصال به اینترنت نیاز خواهند داشت و این کار را برای آن­ها مطلوب­تر خواهد کرد.

در زیر تعاملات بین 3 بخش گفته شده در معماری MVC و نیز ساختاری کلی client/server را می­توانید مشاهده کنید:



همچنین در بخش­های controller و model از معماری layered استفاده شده است. به عبارت دیگر هر یک از دو بخش نام برده شده شامل قسمت­های و componentهایی هستند که componentهایی که در یک حیطه کار می­کنند در یک یک لایه قرار داشته و با دیگر componentها تنها از طریق رابط­هایی ارتباط برقرار می­کنند.

در کل 3 معماری در این سیستم اعمال شده است.

همچنین سیستم را می­توان از 5 دیدگاه متفاوت مشاهده کرد، که عبارتند از Use-Case، Logical، Process، Deployment، implementation و data. در ادامه به توضیح کامل هر یک می­پردازیم.

# 3- اهداف و محدودیت­های معماری

معماری­های نام برده شده که همگی در این سیستم خرید اینترنتی اعمال می­شوند دارای اهداف زیر می­باشد:

* معماری­های مطرح شده (به خصوص حالت کلی client/server) باید قابلیت برقراری ارتباط با دیگر سیستم­ها (مانند سیستم بانک) برای گرفتن خدمات و نیز دادن خدمات به آن­ها را داشته باشد.
* معماری­های مطرح شده نباید موجب کاهش سرعت و کارایی برنامه شوند.
* معماری­ها نباید بر روی کار یکدیگر تداخل و اشکال ایجاد نمایند، بلکه باید در راستای بهبود سیستم حرکت و تعامل نمایند.
* سیستم با چنین معماری­هایی باید دارای امنیت زیاد بوده و اجازه­ی ورود به قسمت­های سرور و نیز پایگاه داده که مهم­ترین بخش­ها به شمار می­روند را به افراد غیر مجاز بدهد.
* سیستم باید بتواند در ساعات مختلف در دسترسی کاربران بوده و نیز کشش تعداد کاربران تا سقف 100.000 کاربر در یک زمان را داشته باشد.
* بنابر آنچه در مورد سرور­های پشتیبان[[1]](#footnote-1) گفته شد (که باید این قابلیت را داشته باشند که بتوانند در صورت بروز مشکل کار سرور دیگر را انجام دهند)، باید سیستم مورد نظر قابلیت جابه­جایی[[2]](#footnote-2) را داشته باشد.
* برای آنکه بتوان پایگاه داده­ای به عنوان پشتیبان داشت و نیز در صورت بالا رفتن درخواست­های از یک پایگاه داده بتوان دیگر درخواست­ها را بر روی پایگاه داده­ی دیگر انتقال داد، باید پایگاه داده­ای که استفاده می­شود بتواند به صورت سنکرون با دیگر پایگاه داده­ها در ارتباط باشد. که این گزینه­های قابل استفاده به عنوان پایگاه داده را کاهش می­دهد.

همچنین چنین سیستمی با محدودیت­هایی نیز همراه است (البته این محدودیت­ها نباید موجب عدم رسیدن به اهداف فوق باشند):

* باید 2 سرور اصلی برای رسیدن به هدف دسترسی در تمام ساعات روز را داشت. زیرا در صورت بروز مشکل در یکی دیگری در مدار وارد شده و بتواند به کاربران سرویس دهد.
* برای بالا بردن امنیت سروری که شامل پایگاه داده است باید از دیگر سرورها جدا باشد. همچنین برای آنکه در صورت بروز مشکل در پایگاه داده اطلاعات از بین نروند باید یک سرور پشتیبان برای سرور پایگاه داده داشته باشیم.

# 4- نمای Use-Case

در این نما بیشتر به بیان سناریویی که در سیستم جریان دارد می­پردازیم.

در حالت کلی، این نما به صورت زیر است:



به عبارت دیگر هر فرد می­تواند آغازگر چندین عمل بوده و نیز یک عمل توسط چندین عامل شروع شوند. همچنین یک عمل نیز می­تواند خود نیز آغازگر عمل دیگر باشد.

عامل­[[3]](#footnote-3)هایی که در این سیستم داریم به شرح زیر هستند:

در اینجا 4 عامل اصلی هستند که می­توانند بر روی سیستم تأثیرگذار باشند. (1) کاربر یا همان مشتری که از سیستم برای خرید استفاده می­تواند. (2) مدیر سیستم که مدیریت کل سیستم بر عهده­ی این شخص است و بیشتری سطح دسترسی را در اختیار دارد. (3) سیستم­های خدمات­رسانی بانک که در قسمت­های مالی سیستم از این خدمات استفاده می­شود. (4) کارکنان فروشگاه اینترنتی که می­توانند کالا اضافه کرده و بسته به سطح دسترسی که دارند بر روی سیستم و برخی از قابلیت­های آن کنترل کنند.

تمام Use-Caseها در سند مدل داده آمده­اند ولی به عنوان نمونه یکی از آن­ها در اینجا آورده شده است:



# 5- نمای منطقی

از دیدگاه این نما زیر سیستم­های زیر را داریم:

* مدیر سیستم (administrator): نمایی که مدیر سیستم مشاهده می­کند، عبارت است از گزارش­گیری از محصولات و نیز کنترل کارمندان و بخش­های مختلف.
* کاربر: نمایی که کاربر مشاهده می­کند، عبارت است از مشاهده­ی محصولات و توضیحات آن­ها. قابلیت دیدن آنچه تا به حال خرید کرده است (history) و نیز صفحه­ی خرید کالا. همچنین کاربر می­تواند سؤالات خود را در سایت نوشته و از سوی کارمندان سایت پاسخ داده شود.
* کارمندان فروشگاه اینترنتی: نمایی که کارمندان مشاهده می­کنند، عبارت است از اضافه کردن کالا و نیز پاسخ به سؤالات و اشکالات کاربران. این کارمندان بسته به سطح دسترسی که دارند می­توانند محیط­های متفاوتی را مشاهده نمایند.

هر یک از این زیر سیستم­ها دارای معماری لایه­ای هستند. در حالت کلی تمامی زیر سیستم­ها دارای لایه­های view، business logic و data access هستند. از طرف دیگر سیستم دارای معماری MVC در سطح بالاتری است. بنابراین هر یک از لایه­هایی که گفته شد در یکی از بخش­های معماری MVC قرار دارند.

## 5-1- بسته­های معماری

معماری سیستم دارای componentهای متفاوتی است که هر یک کار خاص خود را انجام می­دهند. این componentها عبارتند از:

* Authorization و Authentication که برای قسمت­ Authorization در تمام قسمت­های برنامه جریان دارد و استفاده می­شود. (به اصطلاح یک aspect است.)
* Cart که سبد خرید و اعمالی است که با سبد خرید در ارتباط هستند. مثل اضافه کردن به سبد، خریدن کالاهای موجود در سبد.
* Person که شامل تمامی افراد است. این component در سطح پایین­تر خود شامل زیر componentهایی است مثل کارمندان که شامل مدیر سیستم و نیز افراد با سطح دسترسی­های متفاوت و غیر از مدیریت سیستم هستند و نیز کاربران که دید متفاوتی نسبت به کارمندان دارند.
* Database که برای کار کردن و تعامل با پایگاه داده است. برای ارتباط و دادن و گرفتن داده و اطلاعات به/از پایگاه داده تنها باید از طریق این component اقدام کرد.

# 6- نمای فرآیند

سیستم خرید اینترنتی دارای componentهای متعددی است که باید این componentها با هم در تعامل و در ارتباط باشند. این تعامل و ارتباط به کمک دو روش صورت می­پذیرد.

* Message passing
* Method invocation

از طریق این دو روش componentهای مختلف اطلاعات مورد نیاز را بین خودشان تبادل می­کنند.

# 7- نمای فیزیکی

از دیدگاه این نما، برای داشتن سیستمی مناسب و امن نیاز به یک سرور جدا از سرور اصلی برای پایگاه داده­ها نیاز داریم. همچنین دو مورد scalability و availability را نیز می­خواهیم (طبق آنچه در ابتدای سند مورد بحث قرار گرفت). به معنی آنکه سیستم باید این قابلیت را داشته باشد که با افزایش یافتن تعداد کاربران متصل و نیز بالا رفتن درخواست­ها، سرور بتواند بدون بروز هیچ گونه مشکلی کار را پیش ببرد. این مشکل می­تواند از کار افتادن سرور و یا رد درخواست­ها باشد. باید استراتژی مناسبی برای چنین شرایطی در نظر گرفت. ولی یک راه اصلی که باید در هر صورت انجام شود، آن است که می­توان چندین سرور داشت (حداقل دو سرور) که در صورت بالا رفتن load (یعنی بالا رفتن درخواست­های از سرور) برخی از درخواست­ها که بر اساس استراتژی چیده شده مشخص می­شوند به سرور دیگر انتقال یابند. یعنی سرور دیگر در صورت لزوم وارد مدار شده و شروع به سرویس دادن کند. مورد دومی که مورد انتظار است بحث availability است به معنی آنکه سیستم باید بدون مشکلی در هر زمان در دسترسی و آماده­ی خدمت­رسانی باشد (البته این زمان­ها باید مشخص شود و آنکه سیستم در سال و یا در ماه حداکثر چه مدت می­تواند سرویس ندهد). برای رسیدن به این مورد نیز باید می­توان مانند مورد قبل از سرور دیگر استفاده کرد.

همان طور که در قبل گفته شد باید حداقل دو سرور پایگاه داده داشته باشیم زیرا برای آنکه سرعت پاسخگویی به درخواست­هایی که از سرور پایگاه داده می­شود پایین نیاید و نیز در صورت بروز مشکل داده­ها و اطلاعات از دست نروند باید علاوه بر یک سرور پایگاه داده، حداقل یک سرور دیگر به عنوان سرور پشتیبان برای پایگاه داده­ها داشت.

در نتیجه به دو سرور به عنوان سرور اصلی و نیز دو سرور دیگر به عنوان سرور پایگاه داده نیاز داریم.

# 8- نمای پیاده­سازی

همان گونه که در قبل نیز مطرح شد، سیستم دارای معماری MVC در بخش­ها کلی پیاده­سازی است. به عبارت دیگر سیستم به 3 بخش کلی تقسیم­بندی می­شود که هر یک باید بتواند از دو روش مطرح شده در نمای فرآیند با یکدیگر در تعامل باشند.

همچنین بخش­های Model و Controller نیز خود دارای معماری لایه­ای بوده که حداقل دارای دو لایه­ی اصلی Business logic که لایه­ای برای سرویس­رسانی به سیستم­های دیگر نیز به شمار می­رود (البته بر اساس آنکه در کدام یک از بخش­های MVC باشد) و نیز لایه­ی data access که برای ارتباط با پایگاه داده بود و نیز برای کار کردن با داده­ها است. هر لایه ممکن است در هر بخش به صورت سفارشی شده[[4]](#footnote-4) برای آن بخش باشد و نیز ممکن است بر اساس نیاز بخش خود لایه به چندین لایه شکسته شده و یا آنکه با لایه­ای دیگر (به دلیل کارکرد کم) تلفیق شود. البته از بخش­های MVC و نیز معماری لایه­بندی (که دارای لایه­های presentation، business logic و نیز data access است) بخش view تنها در لایه­ی presentation قرار می­گیرد.

اگر بخواهیم دقیق­تر ارتباط این معماری لایه­ای و نیز معماری MVC را شرح دهیم باید به صورت زیر بیان کنیم که:

* بخش Model بیشتر مطابق و متناظر با لایه­ی data access در معماری لایه­بندی است. البته خود بخش model می­تواند دارای چندین لایه باشد که در بالا توضیح داده شد. اعمالی که مرتبط با داده­ها و تعاملات با پایگاه داده هستند در این بخش صورت می­پذیرد.
* بخش Controller بیشتر مطابق و متناظر با لایه­ی business logic است. البته این لایه نیز می­تواند خود شامل چندین لایه­ باشد که هر یک بستگی به عمق و حجم کاری که می­خواهیم در بخش Controller انجام دهیم دارد. درخواست­های آمده از سوی کاربر در این بخش کنترل شده و بنابر درخواست در اینجا تصمیم گرفته می­شود که view و نیز داده­ها به چه صورت تغییر پیدا کنند. به عنوان مثال مشخص می­شود که view بعدی با چه داده­هایی به نمایش در آید.
* بخش View که متناظر با لایه­ی presentation در معماری لایه­بندی است. آنچه به کاربر نشان داده می­شود و کاربر با آن در تعامل مستقیم است در این بخش وجود دارد.

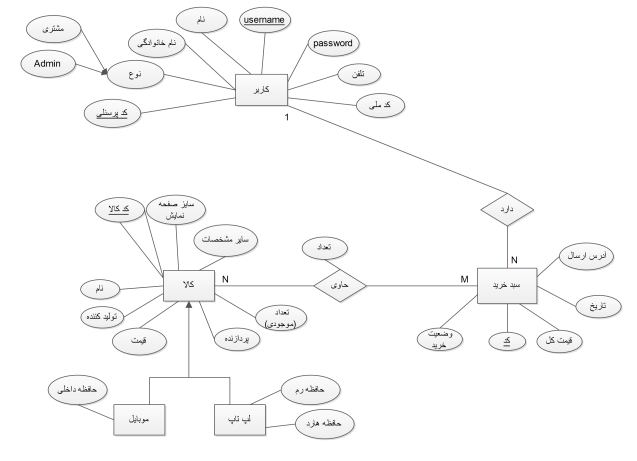
در قبل گفته شد، برای آنکه برای داده­ها و اطلاعات موجود (در پایگاه داده) مشکلی پیش نیاید و نیز بتوان در صورت بالا رفتن درخواست­ها از یک پایگاه داده، برخی از درخواست­ها را به پایگاه داده­ی دیگر منتقل کنیم، نیاز است که نرم­افزار DBMSای که استفاده می­شود قابلیت سنکرون شدن با دیگر سرور­های پایگاه داده را داشته باشد. در این صورت وقتی داده­ای روی یک سرور تغییر می­کند و همان داده از سرور دیگر درخواست می­شود، داده­های نهایی و تغییر داده شده به عنوان پاسخ برگردانده می­شود.

# 9- نمای داده

در سیستم خرید اینترنتی داده­ها و اطلاعاتی وجود دارند که باید در سیستم به صورت پایدار باقی بمانند. بنابراین به پایگاه داده برای نگهداری آن­ها نیاز داریم. موجودیت­هایی که به عنوان موجودیت­های مشهود و اصلی هستند عبارتند از:

* مشتری: کاربری که از سیستم برای خرید استفاده می­کند.
* کارمند: تمام افرادی که در پشت سیستم مسئولیت کنترل کردن بخش­های مختلف را دارند. از مدیر سیستم که دارای بالاترین سطح دسترسی بوده و قابلیت کنترل روی تمامی بخش­ها را دارد تا کارمند عادی که تنها مسئولیت پاسخ به سؤالات مشتری را دارد.
* کالا: جنسی که در فروشگاه اینترنتی برای فروش قرار می­گیرد. این جنس می­تواند دارای اطلاعات مختلفی در ابعاد متفاوتی باشد.
* سبد خرید: سبدی که مشتری (کاربر) اجناسی را که می­خواهد بخرد در آن قرار می­دهد. بدون سبد خرید کاربر قادر به خرید نیست.

نمای داده­ای از پایگاه داده به صورت زیر است (توضیحات بیشتر در سند مدل داده وجود دارد):



# 10- کارایی و اندازه

برای اندازه­گیری و نیز کارایی سیستم معیارهای متعددی وجود دارد که به صورت زیر می­توان تقسیم­بندی کرد:

* اندازه­گیری
  + اندازه­ی کد که نشان­دهنده­ی حجم برنامه است.
  + تعداد کلاس­ها و نیز تعداد formها که به نوعی پیچیدگی برنامه را نیز تعیین می­کنند. همچنین تعداد componentهایی که تولید شده­اند به عنوانی معیار می­توانند باشند.
  + میزان نفر ساعتی که برای تولید پروژه از ابتدا تا انتها وقت گذاشته شد.
  + ...
* کارایی
  + سرعت پاسخ­گویی به درخواست­ها
  + میزان بهره­برداری از CPU سرورها
  + میزان بهره­برداری از حافظه­های سرورها (چه حافظه­های RAM و چه حافظه­های دایمی مانند Hard disk)

# 11- کیفیت

در بررسی کیفیت معیارهای زیر بسیار حائز اهمیت هستند:

* قابل اعتماد: سیستمی که تولید می­شود باید بتواند درخواست­هایی که از وی می­شود را به درستی و با کارایی که برایش تعریف می­شود انجام دهد.
* قابل جابه­جایی: سیستم باید حداقل وابستگی را به سخت­افزار (و نیز در مواردی به نرم­افزار) داشته باشد. زیرا ممکن در صورت بروز مشکلی نیاز باشد که سیستم از سرور فعلی به سرور دیگری منتقل شود (سرور جایگزینی انتخاب گردد)، در این صورت سیستم باید بتواند همانند به خوبی درخواست­ها را پاسخ داده و نیز دچار مشکل نشود.
* امنیت: سیستم باید از لحاظ امنیت در درجه­ای مطلوب و حتی بالاتر قرار بگیرد. به همین منظور سرورهای اصلی که پاسخ­گو به درخواست­های اصلی است از سرور پایگاه داده جدا قرار گرفته­اند. با این روش امکان دسترسی به پایگاه داده کاسته می­شود. همچنین برای جلوگیری از حمله­هایی همچون DOS یک قبل از سرور اصلی قرار گرفته است که در صورت دریافتن حمله از نوع DOS جلوی حمله را می­گیرد.

برای جلوگیری از حمله­هایی به پایگاه داده (یعنی sql injection) کد به صورتی تولید شده است که از اتفاق افتادن چنین حمله­هایی به سیستم کاسته شده است.

* قابلیت دسترسی به سیستم: همانگونه که در قبل مورد بررسی قرار گرفت، سیستم باید این قابلیت را داشته باشد که بتواند در اکثر اوقات در دسترس کاربران برای استفاده از آن باشد. به عبارت دیگر نباید سیستم دچار مشکلی شود که از دسترس کاربران برای استفاده خارج گردد و در صورتی که مشکلی در آن پیش آمد باید یک سیستم جایگزین (که سروری برای این قسمت تعبیه شده است) کار سرویس دادن را انجام دهد.
* قابلیت نگهداری سیستم: یکی از اصول مهم در تولید یک سیستم آن است که بتوان پس از آماده شدن سیستم برای استفاده بتوان از آن نگهداری کرد؛ به معنی آنکه در صورت بروز مشکل و نیز در صورت نیاز به تغییر و یا اضافه کردن یک ویژگی جدید، این کار با حداقل هزینه صورت پذیرد. سیستم خرید اینترنتی در زمان تولید نرم­افزار به صورتی نوشته شده است که قابلیت نگهداری سیستم را پس از تولید و در زمان خدمت­رسانی امکان­پذیر می­کند.

1. Backup [↑](#footnote-ref-1)
2. portability [↑](#footnote-ref-2)
3. actor [↑](#footnote-ref-3)
4. customize [↑](#footnote-ref-4)