



گزارش کارآموزی دوره کارشناسی

رشته

مهندسی کامپیوتر گرایش نرم افزار

موضوع کارآموزی

یادگیری و انجام پروژه بوسیله TDD discipline

نام استاد راهنما

جناب آقای دکتر ایمان عطارزاده

تهیه کننده

960164039

نیکی نجفی

نیمسال دوم سال 99



فهرست

• مقدمه	5
• معرفی کسب و کار	7
• نحوه ی آشنایی با مکان کارآموزی	8
• مشخصات مکان کارآموزی	8
• آشنایی با ورژن 1 اپلیکیشن هم ورزشی	9
• روز و ساعت های حضور در محل کارآموزی	11
خوشبختانه امکان دورکاری در این مجموعه فراهم بوده است و بنده از شنبه تا جمعه هرروز به مدت 6 ساعت فعالیت داشته ام.	11
• TDD (Test Driven Developments)	13
• موجودیت ها (entities)	16
• موارد کاربرد (usecases)	17
• آداپتورهای رابط (interface adapters)	17
• MVC (Model View Controller)	17
○ GUI (Graphical User Interface)	19
• چارچوب ها و درایورها (frameworks and drivers)	19
• Abstraction	20
• تزریق وابستگی (dependency injection)	21
• تزریق سازنده (constructor injection)	21

• (method injection) تزریق متد	22
• Loosely coupled	22
• Tight coupled	22
• (dependency inversion principle) اصل وارونگی وابستگی	22
• (inversion of control) وارونگی کنترل	23
• project based با روش TDD یادگیری فلاتر در	24
• Upresentation	25
• Udomain	26
• Udata	26
• API (Application Programming Interface)	27
• State	27
• Bloc	27
• ایجاد پروژۀ فلاتر	29
• Functional programming	37
• Asynchronous	40
• (getConcreteNumberTrivia()) نوشتن تست برای متد	40
• نحوه ی ران کردن تست	45
• Callable classes	46
• GetRandomNumberTrivia کلاس	51
• Json	56
• Serialization	56
• Deserialization	57

• swagger	59
• swagger کردن در authorize نحوه ی	61
• postman کردن در authorize نحوه ی	61
• authentication و authorization فرق	63
• کردن autogenerate نحوه ی	64
• Udata لایه	65
• مدل ها (models)	66
• Factory	72
• Remote data source	77
• Local Data Source	80
• http متداول ترین ارورهای	81
• HTTP Error 401 (Unauthorized)	82
• HTTP Error 400 (Bad Request)	83
• HTTP Error 404 (Not Found)	83
• HTTP Error 500 (Internal Server Error)	83
• پیشنهادات	84
• خلاصه کارنهایی	84
• کارهای آتی	85
• منابع	86

● مقدمه

بنده در حدود شش ماه کارآموز شرکت نوآوران ورزش دیجیتال بوده ام و گزارش زیر حاصل دوره 3 ماهه بنده نزد این شرکت میباشد.

این شرکت قصد توسعه اپلیکیشنی به نام هم ورزشی را دارند که این اپلیکیشن با زبان فلاتر نوشته شده است که بنده در 3 ماه اول آنرا فراگرفتم و قرار است فناوری های توسعه آن ارتقا پیدا کند و از دیسپلین TDD (Test Driven Development) استفاده شود که گزارش بنده درباره ی یادگیری این دیسپلین از طریق project-based است.

بنده در این دوره کارآموزی بسیار آموختم با تشکر است استاد راهنمای بنده جناب آقای دکتر ایمان عطارزاده و جناب آقای مهدی قلیزاده که در این امر تلاش بسیار زیادی کرده اند.

فصل اول : آشنایی کلی با مکان کارآموزی

• معرفی کسب و کار

شرکت نوآوران ورزش دیجیتال در زمینه هایی مانند خرید ، فروش ، توزیع ، تولید ، بسته بندی ، واردات و صادرات و خدمات پس از فروش کلیه ی کالاهای مجاز بازرگانی ، البسه و کلیه ی کالاهای ورزشی و انواع مکمل های مجاز ورزشی و... فعالیت دارد این شرکت در تاریخ 1399/2/14 تاسیس شده است و قصد راه اندازی اپلیکیشن همورزشی با استفاده از دسیپلین TDD است که در حال حاضر نسخه ی ورژن 1 بالا بوده و برای راه اندازی نسخه ورژن 2 که با زبان فلاتر و دسیپلین TDD نوشته شده است تلاش میکنیم این اپلیکیشن 2 نوع کاربر دارد که یکی مشتری باشگاه و دیگری صاحب باشگاه میباشد که مشتری با انتخاب کردن باشگاه مورد نظر خود و انتخاب بسته های مورد نظر میتواند از امکانات باشگاه ها برخوردار شود.

گزارش زیر پروژه ای آزمایشی برای یادگیری TDD میباشد.

• نحوه ی آشنایی با مکان کارآموزی

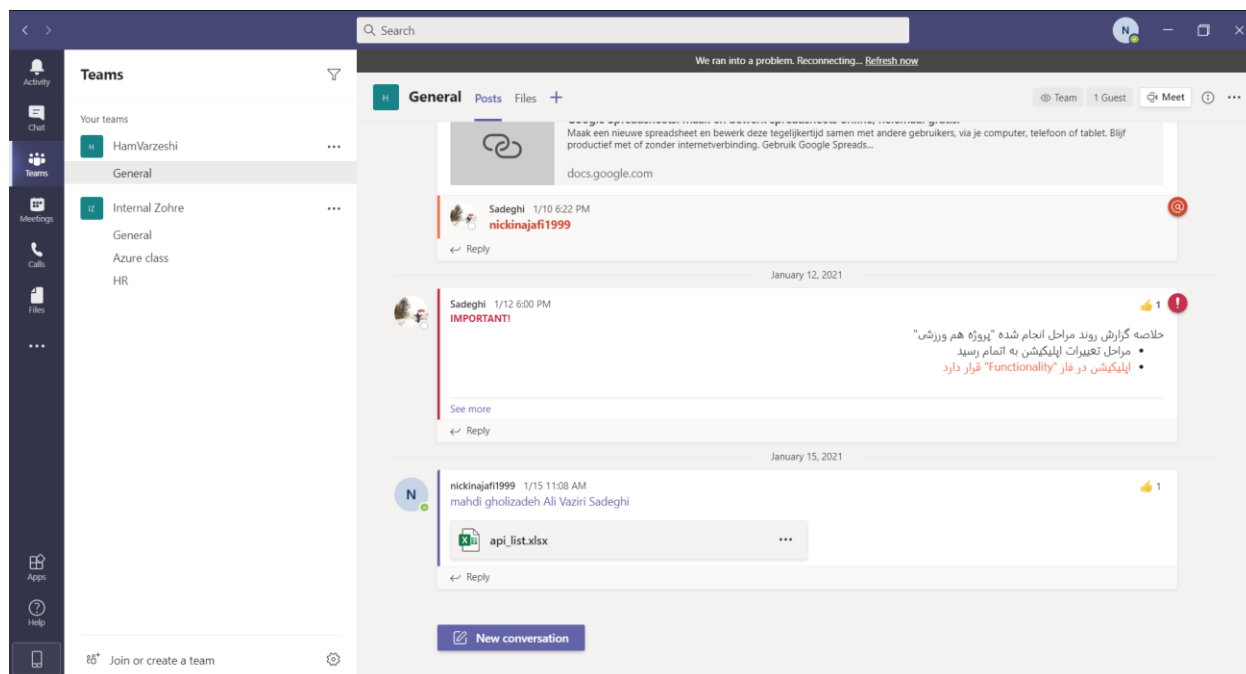
بنده به مدت 1 الی 2 ماه به دنبال کارآموزی بودم و به خاطر پاندمی کرونا اغلب با شرایط خاصی کارآموز قبول میکردند و با پرس و جو از دانشجویان و افراد مختلف در نهایت توانستم مصاحبه ای را با شرکت نوآوران ورزش دیجیتال داشته باشم که در این مصاحبه به عنوان کارآموز برنامه نویسی اپلیکیشن با فلاتر برگزیده شدم.

• مشخصات مکان کارآموزی

این شرکت واقع در استان تهران – منطقه 14 ، شهرستان تهران ، بخش مرکزی ، شهر تهران، محله تیموری ، بلوار شهیدجواد اکبری ، خیابان شهید احمد قاسمی ، پلاک – 68 ، طبقه چهارم ، واحد 7 میباشد.

آقای مهدی قلیزاده مسئول برنامه نویسی اپلیکیشن میباشد. در این دوره آقای مهدی قلیزاده با نظارت بسیار دقیق تمامی deadline ها و تمام کارهای انجام شده را بررسی و گزارش تهیه میکردند که تمامی گزارشات و

وضعیت اپلیکیشن در محیط Microsoft teams قرار گرفته است.



• آشنایی با ورژن 1 اپلیکیشن هم ورزشی

قبل از کارآموزی در این مجموعه مشغول به برنامه نویسی
front-end وبسایت بودم و با
html,css,javascript,jquery,bootstrap,materializecss,photoshop,git
و کمی php نیز کار کرده بودم و بسیار علاقه مند به
یادگیری برنامه نویسی سمت اپلیکیشن نیز بودم که با زبان
فلاتر آشنا شدم.

از این زبان حتی برای توسعه ی وبسایت نیز استفاده میشود و بر خلاف جاوا و بسیاری از زبانهای برنامه نویسی دیگر از این زبان هم برای توسعه اپلیکیشن برای اندروید و هم برای iOS استفاده میشود که جز بهترین ویژگی های فلاتر محسوب میشود. فلاتر بر روی زبان دارت پیاده سازی شده است و زبان نسبتا جدیدی میباشد و در حال بهبود یافتن است از بدی های آن میتوان به این اشاره کرد که نسبت به زبان java، community بزرگی ندارد و به همین خاطر اکثر اوقات که به مشکل میخوردم و در بستر وب به دنبال پاسخ بودم جوابی نمیافتم و حتی گاهی issue جدید در گیت هاب باز میکردم به امید اینکه کسی پاسخگو باشد و سعی میکردم در صورتی که پاسخی نمیافتم از جناب آقای مهدی قلیزاده درخواست کمک میکردم و ایشان به بهترین نحو ممکن راهنمایی میکردند.

همانطور که اشاره شد این اپلیکیشن با استفاده از فلاتر و همچنین RESTAPI توسعه داده شده که بنده قرار بود این اپلیکیشن را بهبود داده و ورژن جدیدی را release کنم برای این کار لازم بود تمامی کدها و داکيومنتیشن های ورژن 1 را خوانده و نحوه ی functionality اپلیکیشن را متوجه

شوم که کاری وقت گیر و گاه نیز سرسام آور بود ورژن 2 قرار است که از graphql و همچنین دسیپلین TDD استفاده کند همچنین تغییراتی در UI نیز لحاظ شده است گزارش زیر پروژه ای آزمایشی برای یادگیری TDD میباشد.

• روز و ساعت های حضور در محل کارآموزی

خوشبختانه امکان دورکاری در این مجموعه فراهم بوده است و بنده از شنبه تا جمعه هرروز به مدت 6 ساعت فعالیت داشته ام.

فصل دوم : ارزیابی بخش های مرتبط با رشته ی علمی کاراموز

• TDD (Test Driven Developments)

Tdd یعنی ابتدا تست (test) برنامه را قبل از پیاده سازی (implementation) بنویسیم

3 قانون در این امر وجود دارد که در زیر به

آن اشاره میکنم :

1. تنها در صورتی کد پیاده سازی رو مینویسیم که
بخوایم یک واحد در حالت fail رو به حالت pass
برسونیم.

2. مجاز به نوشتن بیش از یک آزمون واحد نیستیم. و
failure ، compilation failure است.

3. اجازه نوشتن production code بیشتر از مقدار
نیاز برای تغییر وضعیت unit test از fail به pass را
نداریم.

Unit test را برای عملکرد هایی (functionality) که
میخوایم توی کد بیاوریم مینویسیم.

با توجه به قانون دوم در صورتی که unit test ما
شکست (fail) خورد باید شروع کنیم و کدهای پیاده
سازی (production code) را بنویسیم. با توجه به

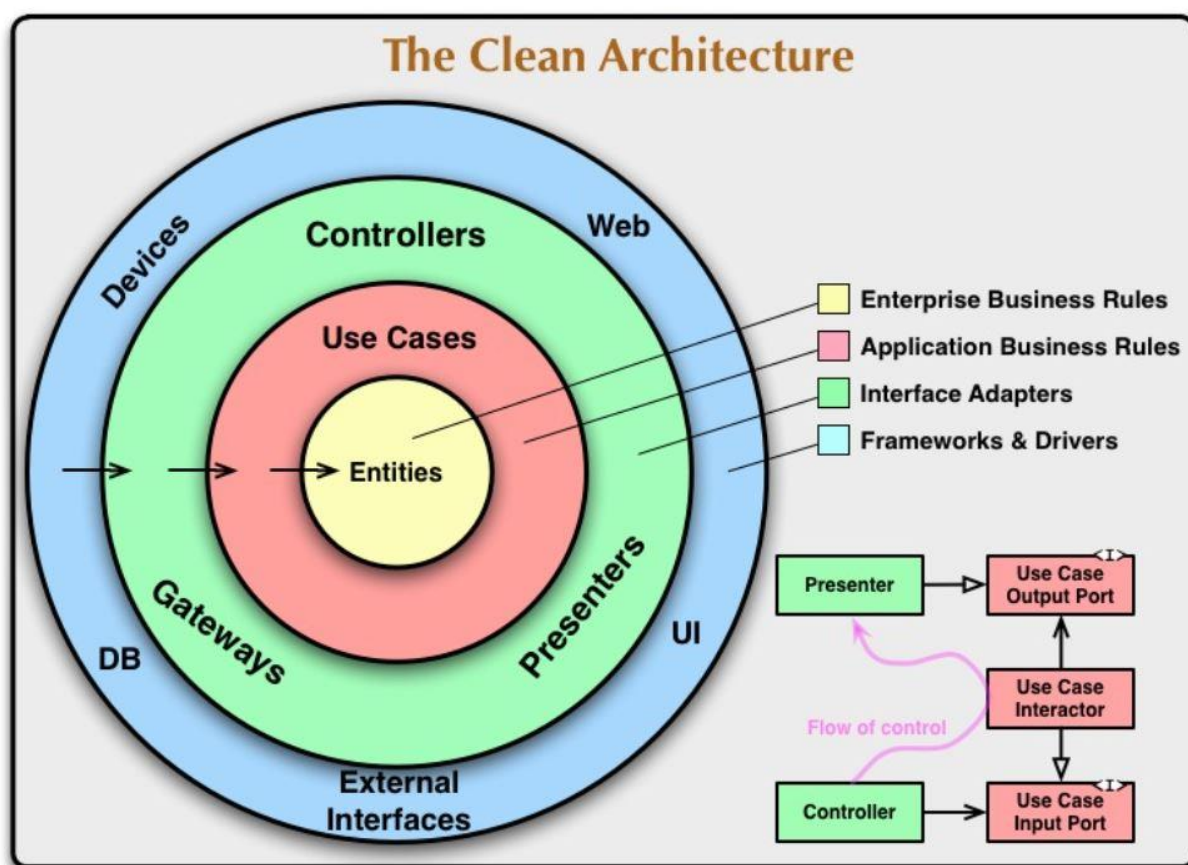
قانون سوم تنها در صورتی کدهای پیاده سازی (production code) مینویسیم که test ما به حالت کامپایل (compile) و یا به وضعیت (pass) تغییر پیدا کند.

برای اینکه متوجه بشویم که برای چه از TDD استفاده میکنیم این مثال را در نظر بگیرید که در طول روز چقدر وقت و انرژی برای debug کردن میگذارید. با استفاده از tdd که نوعی discipline است این زمان به مقدار بسیار زیادی کاهش پیدا میکند.

دلیل اینکه کد بهم ریخته و نامرتب رو مرتب نمیکنیم این است که میترسیم آنرا خراب کنیم ولی با استفاده از تست ها میفهمیم که کدمان سالم است و در صورت بروز خرابی و مشاهده کد نامرتب بلافاصله میتوانیم کد را درست کنیم و به راحتی تمام تغییرات دلخواه را اعمال کنیم. در واقع تست ها به ما نشان میدهند که سیستم چگونه کار میکند و ما نیز چگونه باید با سیستم کار کنیم. اضافه کردن تست به سیستم هایی که بدون تست طراحی شده اند کار بسیار بسیار سختی است زیرا کد آنها برای تست طراحی نشده اند. به قابلیت تست کردن (testable)

جداشده (decoupled) نیز میگویند زیرا برای تست کردن ماژولی به تنهایی باید ابتدا آنرا از سایرین جدا کنیم.

در کل کاری که میکنیم این است که تستی را بنویسیم که fail شود سپس به اندازه ای کد implementation را بنویسیم که آن تست به حالت pass در بیاید.



دایره های متحد المركز بالا قسمت های مختلف نرم افزار را

نمایش می‌دهند قسمت‌های داخلی سطح بالاتر هستند.
دایره‌های بیرونی مکانیسم‌ها (mechanisms) هستند.
دایره‌های داخلی سیاست‌ها (policies) هستند. حلقه
های داخلی راجع به حلقه‌های بیرونی نمیتوانند چیزی را
بدانند. تمام کلاس‌ها (classes)، متغیرها (variables)،
tokens، functions و ... که در دایره‌های بیرونی
هستند نباید به آنها در دایره‌های درونی اشاره شود.

• موجودیت‌ها (entities)

موجودیت‌ها میتوانند function، datastructures، ...
باشند وقتی در لایه‌های خارجی تغییری ایجاد میشود به احتمال
خیلی زیادی چیزی در این لایه تغییر نمیکند به عنوان مثال
هنگامی که navigation یا security در صفحات تغییر
میکند احتمال اینکه موجودیت‌ها تغییر کنند بسیار کم
است.

- موارد کاربرد (usecases)

در واقع موارد کاربرد جریان دیتا (dataflow) از موجودیت ها و به آنها را کنترل میکنند. تغییرات در این لایه بر روی لایه ی موجودیت ها تاثیری ندارد ولی تغییرات عملکرد کلی نرم افزار تغییراتی را در این لایه ایجاد میکند.

- آداپتورهای رابط (interface adapters)

در این لایه داده ها از فرمتشان در usecases , entities به فرمتی جدید در web یا database یا عوامل خارجی دیگری تبدیل میشوند. این لایه در واقع معماری MVC از GUI را در بر دارد. مدل ها از طریق کنترلر ها به موارد کاربرد داده میشوند و سپس از طریق usecase ها به view و presenters داده میشوند.

- MVC (Model View Controller)

الگوی طراحی نرم افزار است که از 3 قسمت تشکیل شده است.

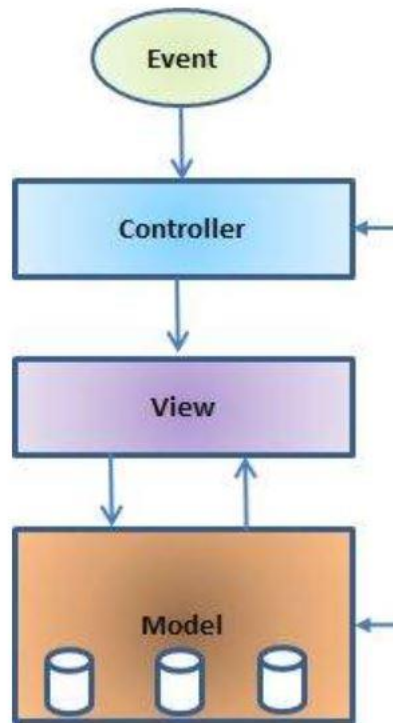
Model : پایین سطح ترین قسمت در این الگو که کار نگهداری و مدیریت دیتا بر عهده اوست همچنین به درخواست های view پاسخ میدهد و به دستورالعمل های controller نیز عکس العمل نشان میدهد تا خود را بروز رسانی کند.

View : script-based هستند قسمتی و یا همه ی دیتاها را در فرمت مورد نظر نمایش میدهد.

Controller : تعاملات بین model و view را کنترل میکند همچنین ورودی های کاربر را نیز اعتبارسنجی میکند.

دلیل مشهور بودن این الگو این است که قسمت منطق برنامه از قسمت ال جدا شده است.

قسمت controller تمام request ها را دریافت میکند و با کمک model دیتاهای مورد نیاز view را فراهم میکنند سپس view دیتاها را به نمایش میگذارد.



نمایش الگوی MVC

○ GUI (Graphical User Interface)

با استفاده از موس و ... به کاربر اجازه میدهد تا با برنامه در تعامل باشد. در GUI تمام تعاملات از طریق کاربر با رابط کاربری برنامه است.

• چارچوب ها و درایورها (frameworks and drivers)

دیتابیس (database) و وب فریم ورک ها (web frameworks) جزئیات محسوب میشوند که در بیرونی

ترین لایه که کمترین آسیب را درست میکنند قرار دارند در این قسمت تقریبا هیچ کدی نوشته نمیشود.

اینکه حتما 4 تا لایه داشته باشیم قانون کلی است ولی ممکن است کسی در معماری نرم افزارش بیشتر از این 4 لایه را داشته باشد. هر چه به لایه های درونی نزدیکتر میشویم abstraction افزایش میابد.

• Abstraction

برای بالابردن کارایی و کاهش پیچیدگی نرم افزار استفاده میشود که در آن برنامه نویس تنها دادهای مرتبط برای object را نمایش میدهد.

در شکل جریان کنترل را در سمت راست پایین میتوانیم مشاهده کنیم که از controller شروع شده از usecase میگذرد و در نهایت به presenter ختم میشود.

فرض کنید که usecase میخواهد presenter را صدا کند طبق قانون اشاره شده از ... , variables , function در لایه های بیرونی نمیتوانیم در لایه های درونی استفاده کنیم.

راه حل این است که usecase یک usecase interface (output port را call کرده سپس در لایه ی بیرونی یعنی presenter آنرا پیاده سازی کنیم. در اینجا interface باعث جداسازی (seperation) میشود.

• تزریق وابستگی (dependency injection)

یک الگوی طراحی است که یعنی با استفاده از یک رابط و یا (interface) وابستگی میان 2 کلاس را حذف کنیم. وابستگی شدید 2 کلاس به هم دیگر (tight coupled) باعث میشود که انعطاف پذیری نرم افزار به شدت پایین بیاید. در این روش ماژول های سطح پایین به ماژول های سطح بالا که شامل انتزاع ها هستند تزریق میشوند. تزریق وابستگی از 2 طریق (تزریق سازنده – تزریق متد) انجام میشود)

• تزریق سازنده (constructor injection)

در این روش شیای از کلاس سطح پایین به سازنده کلاس سطح بالا ارسال میشود که در نهایت نوع آن از جنس Interface است.

- تزریق متد (method injection)

حال اگر بخواهیم تنها یک متد مشخص از یک کلاس را درگیر بحث تزریق وابستگی کنیم باید به چه صورت عمل کرد؟ باید تنها و تنها آن وابستگی را به یک آرگومان یک متد مشخص ارسال کنیم. بنابراین تزریق متد تنها و تنها روی یک متد مشخص صورت میگیرد و طی آن وابستگی‌های کلاس‌های سطح پایین به متدی از کلاس سطح بالا ارسال می‌شود

- Loosely coupled

بدین معنی‌ست که کمترین وابستگی بین دو کلاس وجود داشته باشد.

- Tight coupled

بدین معنی‌ست که بیشترین وابستگی بین دو کلاس وجود داشته باشد.

- اصل وارونگی وابستگی (dependency inversion principle)

اصل وارونگی وابستگی یک اصل طراحی نرم افزار است که به ما در تولید یک نرم افزار با استحکام ارتباطی کم

(Loosely Coupled) کمک می کند. بر اساس این

تعریف اصول وارونگی وابستگی عبارتند از:

1. ماژول های سطح بالا نباید به ماژول های سطح پایین وابسته باشند بلکه هر دو باید به یک رابط (interface) متصل شوند.

2. انتزاع (abstraction) نباید به جزئیات وابسته باشند. جزئیات باید به انتزاع وابسته باشند. (منظور از انتزاع دید کلی نسبت به یک شیء است مثلاً وقتی ما می‌گوییم میز چیزی که در ذهن ما نقش می‌بندد یک شکل کلی است ولی وقتی می‌گوییم میز ناهارخوری دقیقاً مشخص می‌کنیم که چه نوع میزی است. در نتیجه انتزاع یک دید کلی از یک شیء بحساب می‌آید)

• وارونگی کنترل (inversion of control)

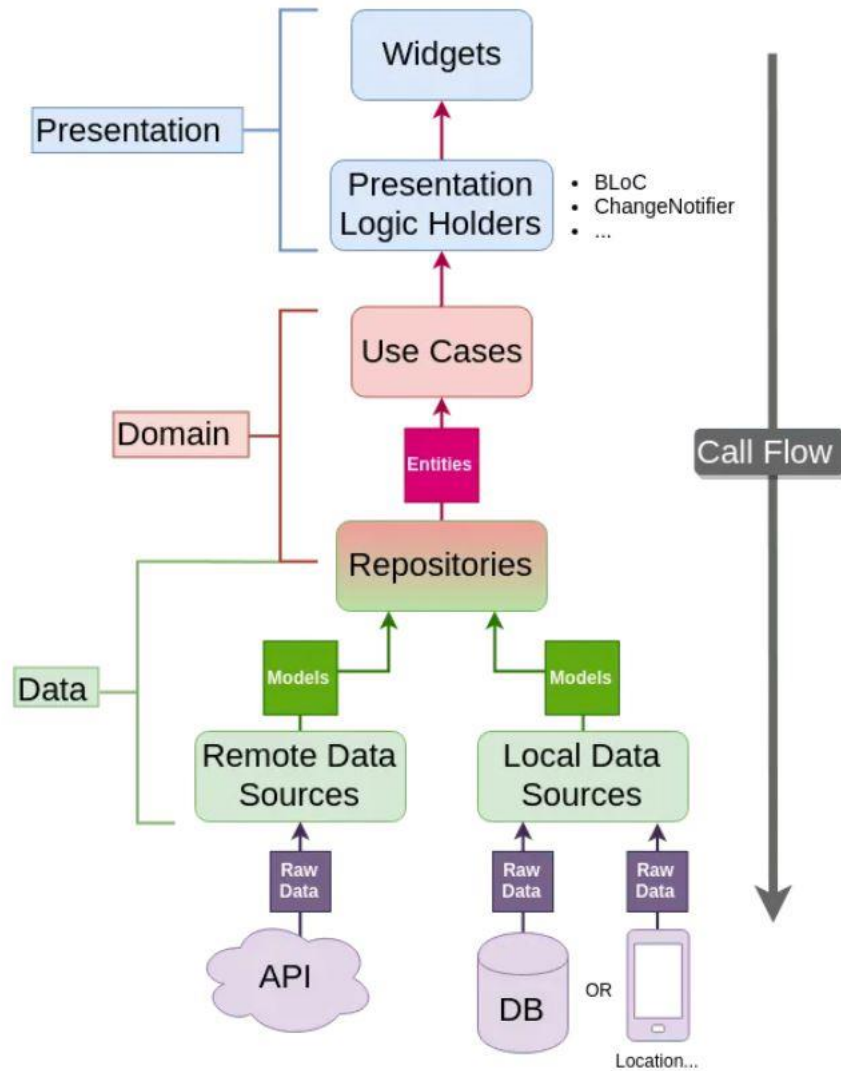
در این روز بجای اینکه ماژول های سطح بالا را به ماژول های سطح پایین متصل کنیم به abstraction متصل میکنیم. بنابراین باید یک رابط (interface) ایجاد کنیم که در آن یک انتزاع (abstraction) تعریف کنیم.

• یادگیری فلاتر در TDD با روش project based

در اینجا تنها از طریق کد زدن میتوانیم واقعا این روش را یاد بگیریم. نام اپلیکیشنی که میخواهیم پیاده سازی کنیم Number Trivia App میباشد که قرار است در این اپلیکیشن هر عددی که وارد میشود تکستی در زیر آن نمایش داده شود.

در این اپلیکیشن میخواهیم کارهایی مانند گرفتن دیتا از API یا حافظه محلی (local cache) ، مدیریت خطا (error handling) ، اعبارسنجی ورودی ها (input validation) و ... را انجام بدهیم. برای state management از bloc استفاده میکنیم.

در این اپلیکیشن ویژگی (feature) داریم مثلا گرفتن تکست های اعدادی که کاربر وارد میکند. که همانطور که در شکل زیر مشاهده میکنید این ویژگی ها به 3 لایه تقسیم میشوند , data , domain , presentation.



• لایه presentation

در این لایه برای نمایش دادن هرچیزی از widget ها استفاده میکنیم که این ویجت ها event ها را به bloc میفرستند و حواشان به state ها نیز هست.

• لایه domain

این لایه شامل usecases و entity ها است و کاملاً مستقل از سایر لایه ها می باشد. Repository هم در لایه ی دیتا و هم در لایه ی domain می باشد. که این امر از طریق dependency inversion که در بالا توضیح داده شد امکان پذیر می باشد یعنی repository یک کلاس abstract است که کارهایی که باید انجام شود در اینجا توضیح داده میشود که در لایه ی domain قرار میگیرد و قسمت پیاده سازی (implementation) در لایه ی دیتا قرار میگیرد. Entity در این اپلیکیشن عدد و تکست نمایش داده شده تعریف میشود.

• لایه data

این لایه از repository ، datasources (که شامل گرفتن api و همچنین cache کردن آن میشود) در این لایه repository مشخص میکند که آیا دادهای cache شده را نشان بدهیم ؟ اگر بله کی دیتا ها را cache کنیم ؟ و در remote datasource ما از HTTP GET REQUEST در api استفاده میکنیم

و در local datasource از shared preferences برای cache کردن داده‌ها استفاده می‌کنیم که هر دو بصورت ترکیب شده در repository قرار دارند. در این اپلیکیشن در صورتی که اتصال اینترنت وجود دارد داده‌ها را از api گرفته سپس آنها را cache می‌کنیم و در صورتی که اتصال اینترنت نداشته باشیم از آخرین داده‌ی cache شده استفاده می‌کنیم.

• API (Application Programming Interface)

وظیفه‌ی جابجایی دیتا از نقطه‌ای به نقطه دیگر از صفحه یا لینکی به صفحه یا لینکی دیگر بر عهده‌ی API می‌باشد.

• State

داده‌ای که ویجت پس از رندر و یا ریفرش شدن دارد.

• Bloc

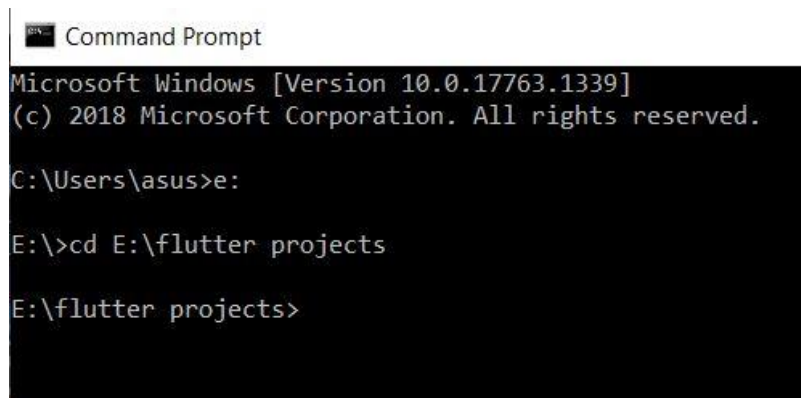
کتابخانه‌ی state management است که قرار است لایه‌های presentation را از لایه‌های منطقی جدا کند همچنین استفاده از این کتابخانه اضافه کردن قابلیت تست را بسیار آسان می‌کند.

فصل سوم : آزمون آموخته ها ، نتایج و پیشنهادات

در این مرحله می‌خواهیم قسمت هایی که تا به الان یاد گرفته ایم را به عمل تبدیل کنیم.

• ایجاد پروژه فلاتر

ابتدا cmd را باز کرده و فولدري را که می‌خواهیم پروژه را در آن تشکیل بدهیم را با استفاده از دستور زیر انتخاب می‌کنیم.



```
Command Prompt
Microsoft Windows [Version 10.0.17763.1339]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\asus>e:

E:\>cd E:\flutter projects











E:\flutter projects>
```

سپس با استفاده از دستورات زیر پروژه فلاتر جدیدی را در این فولدر ایجاد می‌کنیم.

```
E:\flutter projects>flutter create number_trivia
```

در اینجا number_trivia نام فولدري است که پروژه در آن تشکیل میشود.

پس از اجرای این دستور فایل‌هایی به ساختار زیر در آدرس مقصد تشکیل میشوند.

 .dart_tool	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	File folder	
 .idea	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	File folder	
 android	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	File folder	
 ios	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	File folder	
 lib	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	File folder	
 test	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	File folder	
 .gitignore	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	Text Document	1 KB
 .metadata	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	METADATA File	1 KB
 .packages	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	PACKAGES File	3 KB
 number_trivia.iml	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	IML File	1 KB
 pubspec.lock	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	LOCK File	4 KB
 pubspec.yaml	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	YAML File	3 KB
 README.md	۱۳۹۹/۰۹/۱۴ ب ۰۶:۵۵...	Markdown File	1 KB

در فولدر `lib` فولدری دیگر به نام `features` میسازیم که در هر یک از `feature` هایی که در این فولدر تعریف میکنیم ۳ لایه قرار دارند. , `Login` همه ی اینها `feature` محسوب میشوند منتهی ما در اپلیکیشنمان تنها `feature` یی که داریم `number_trivia` میباشد. همانطور که در بالا اشاره کردیم داخل هر یک از `feature` ها 3 لایه قرار میگیرند (`data,presentation,domain`).

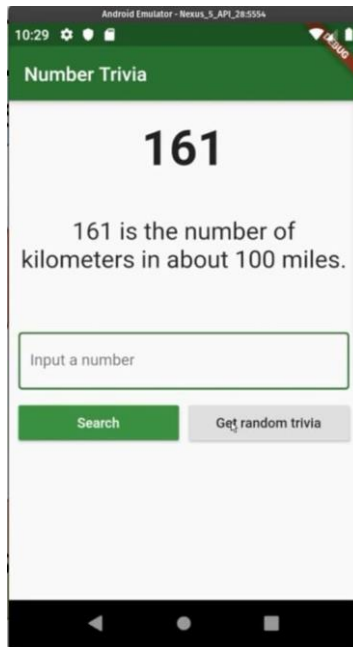
در فولدر `lib` فولدر دیگری به نام `core` داریم که تمام `function` ها و تمام ویژگی هایی که چندین فایل با هم به

اشتراک دارند همگی در این قسمت قرار دارند. تا این مرحله به بالاترین سطح معماری دست یافته ایم.

presentation logic holders برابر با state management میباشد.

در فولدر presentation ۳ فولدر دیگر خواهیم داشت به نامهای bloc (که در واقع همان state management ما است)، widgets، pages این دو فولدر از هم مجزا هستند زیرا نمیخواهیم کدهای یوآیمن بیش از حد در یک فایل قرار بگیرند.

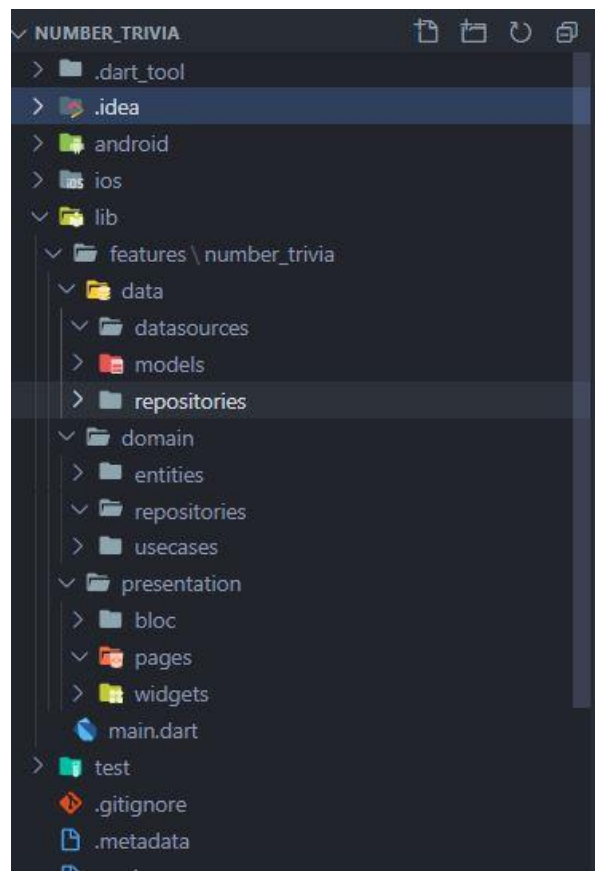
Usecase ها در اینجا در واقع کلاس هستند.



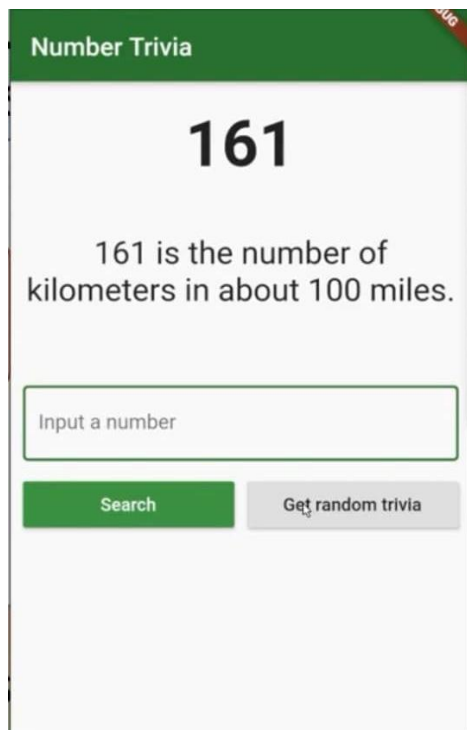
در عکس بالا اپلیکیشن نهایی نمایش داده میشود که در اینجا ۲ کلاس داریم یکی `get_concrete_number_trivia()` که در صورتی که روی `search` کلیک کنیم اجرا میشود و دیگری `get_random_number_trivia()` که در صورتی که روی `get random trivia` کلیک میکنیم اجرا میشود.

در لایه ۳ `domain` فولدر به نامهای `entities` (همان عدد و تکست نمایش داده شده در اپلیکیشن میباشد) `repositories` (که در واقع همان `contract` و یا `abstract` کلاس برای `repository` در این قسمت قرار میگیرد) و در نهایت `usecase` ها نیز در این فولدر قرار میگیرند.

در لایه ی data نیز 3 فولدر وجود دارد , datasources , models , repositories
در زیر معماریهایی که در بالا توضیح داده شد نمایش داده میشود.



اولین قدم برای تولید یک اپلیکیشن ایجاد طراحی UX , UI
آن است. که در زیر نمایش داده شده است.

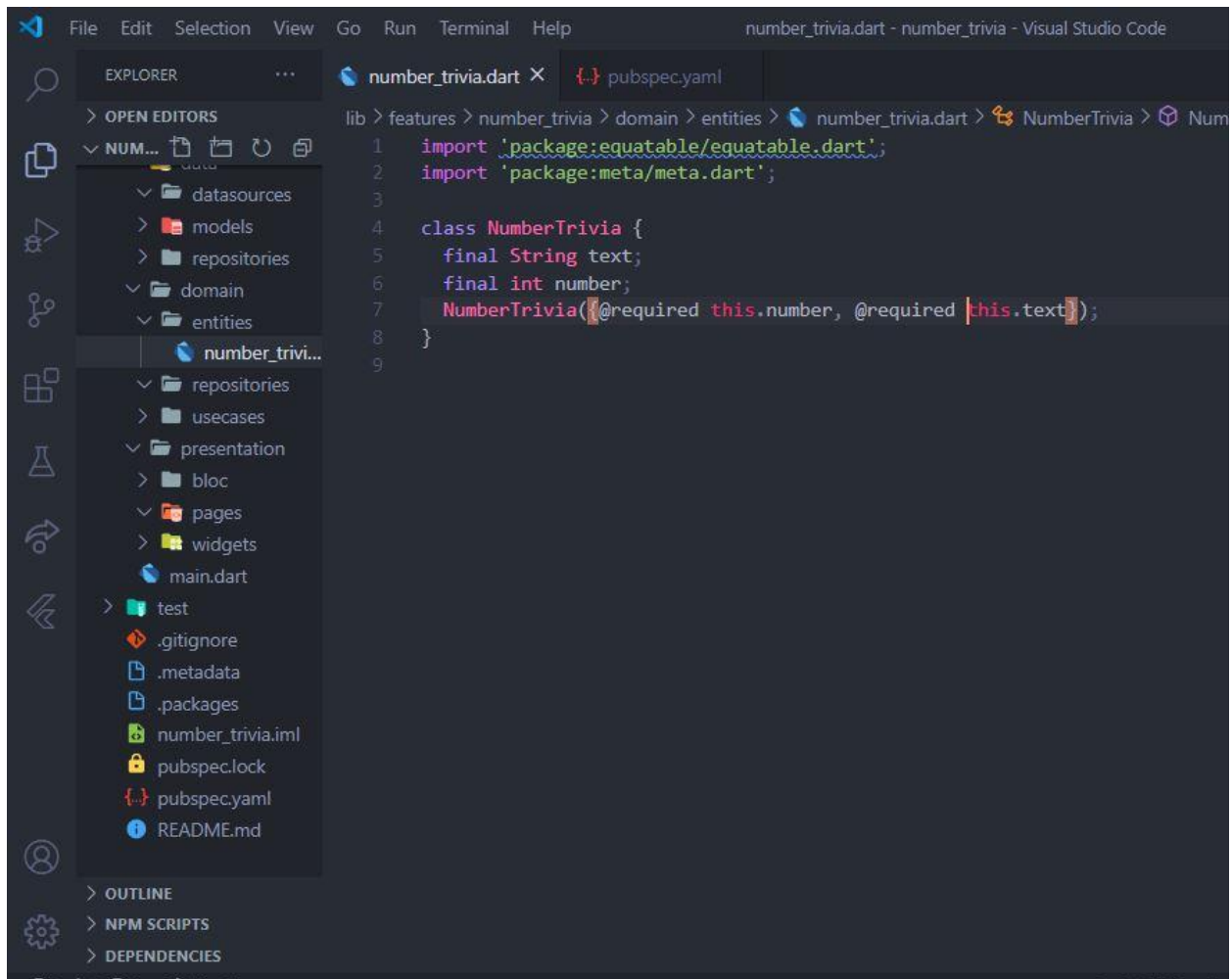


پروسه کدینگ از داخلترین و ثابت ترین لایه شروع میشود
که در اینجا یعنی entities در لایه domain برای اینکه
بفهمیم چه فیلدهایی در entities قرار میگیرند به
ریسپانسهایی که از طرف random_number , concrete
number میایند توجه میکنیم. مثلاً در گرفتن داده ها از
طریق random_number برای عدد 42 از url زیر استفاده
میکنیم :

<http://numbersapi.com/42?json>

```
{  
  "text": "42 is the number of spots (or pips, circular patches or pits) on a pair of standard six-sided dice.",  
  "number": 42,  
  "found": true,  
  "type": "trivia"  
}
```

در این قسمت text و number قسمت هایی هستند که نیاز داریم و در صورتی که عددی وجود نداشته باشد مقدار false : "found" میشود و type در اینجا همیشه trivia میباشد پس چون ثابت است به آن نیازی نخواهیم داشت برای فایل number_trivia.dart تستی ننویسیم زیرا اصلا چیزی وجود ندارد که بخوایم تست کنیم از پکیج eduatable برای مقایسه مقداری آجکت ها استفاده میکنیم. این فایل را در زیر مشاهده میکنید :



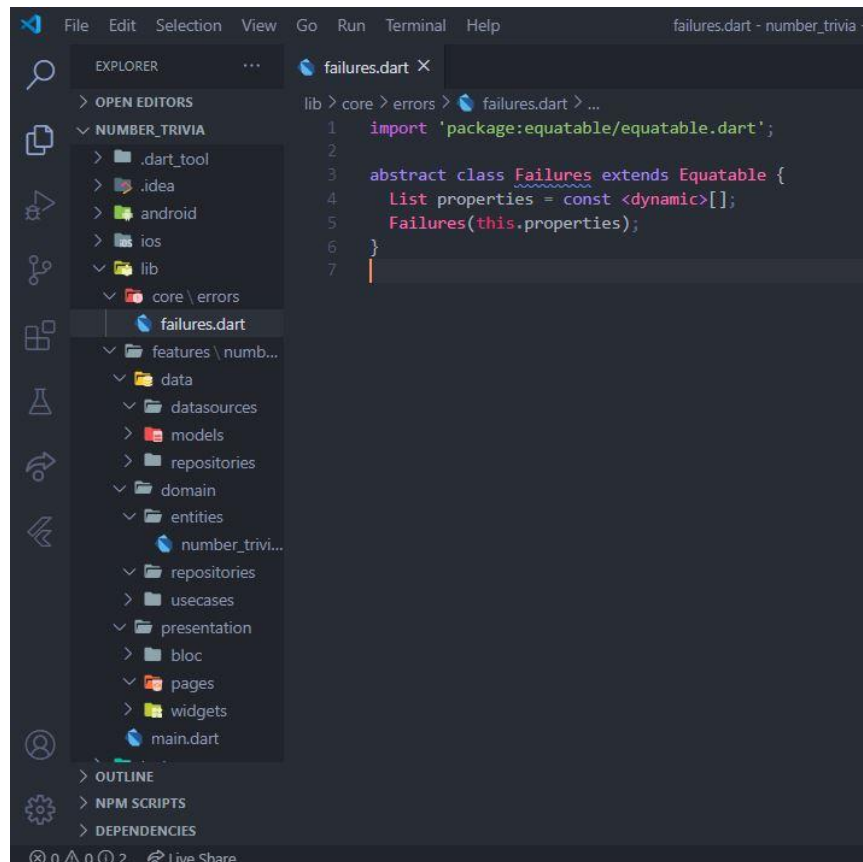
Usecases دیتا ها را از repository میگیرند. در اینجا 2
نوع از آنها را خواهیم داشت
GetConcreteNumberTrivia ,
.GetRandomNumberTrivia
Usecases داده ها را از entities از طریق repositories
میگیرند و سپس آنها را به لایه ی presentation ارسال

میکنند. هر دوی repositories , usecases و هم numbertrivia و هم failure برای error handling را بر میگردانند اما چگونه؟ از طریق functional programming

• Functional programming

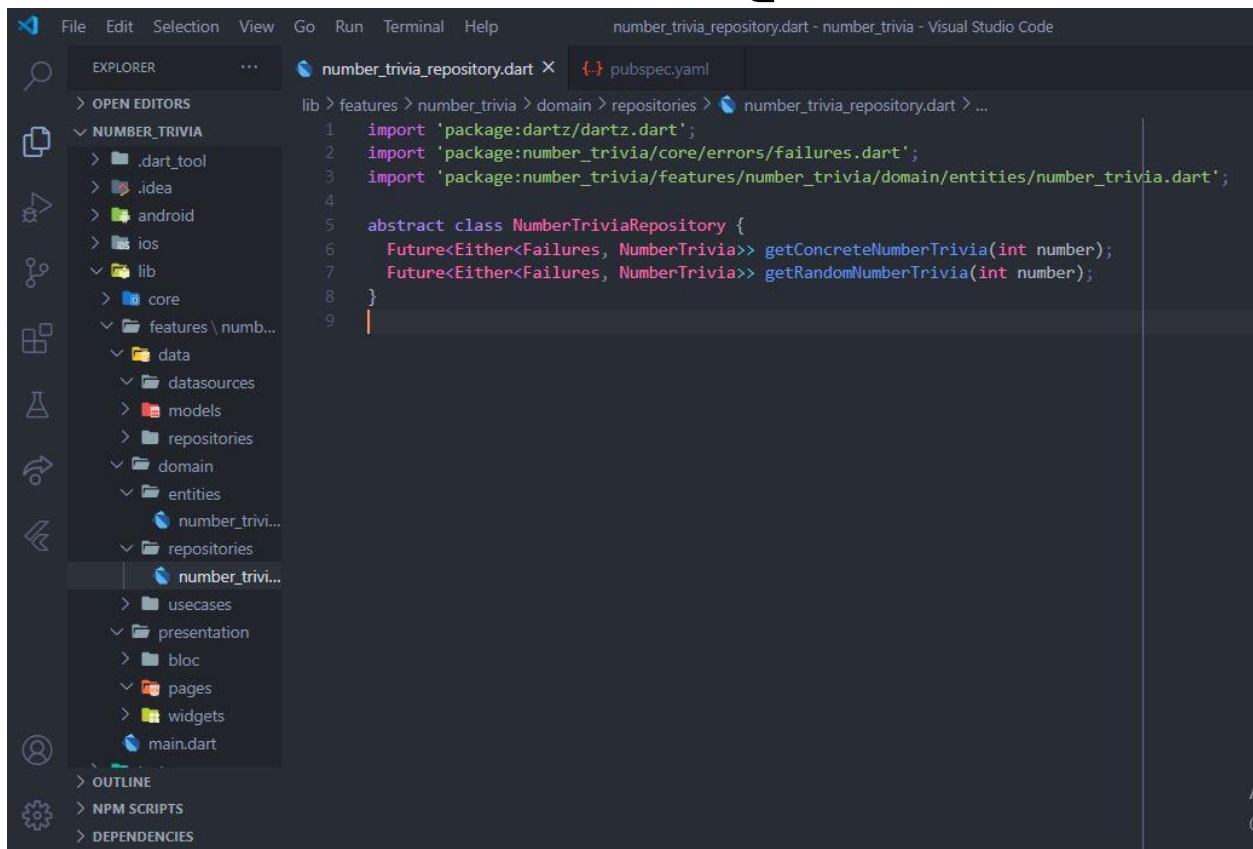
در آن فانکشن ها درخت هایی از عبارات هستند که هر کدام از آنها مقداری را برمیگردانند.

پکیج dartz قابلیت functional programming را اضافه میکند که میتوان در آن از $\text{Either}\langle L, R \rangle$ استفاده کرد که در آن $L = \text{Failure}$ و $R = \text{NumberTrivia}$ است. Failures در بسیاری از لایه های اپلیکیشن ما استفاده میشود پس آنرا در فولدر core قرار میدهیم. در اینجا failure نوعی abstract class میباشد که سایر failure ها مانند server failure و ... از آن حاصل میشوند. در اینجا از قابلیت های پکیج equatable استفاده میکنیم زیرا میتوانیم آبجکت ها را با استفاده از این پکیج باهم مقایسه کنیم. کد مربوط به فایل failures.dart را در زیر مشاهده میکنید :



تست کردن بدون پیاده سازی در فلاتر از طریق پکیج mockito امکان پذیر میباشد. الان میخواهیم تعریف یا همان قسمت contract از repositories را پیاده سازی کنیم که در قسمت domain قرار دارد. در آن 2 متد را داریم ، `getConcreteNumberTrivia()` ، `getRandomNumberTrivia()` که در آنها `Future<Either<Failure , NumberTrivia>>` میباشد در متد `getConcreteNumberTrivia` کاربر عددی را وارد میکند که به این فانکشن داده میشود. type ای که برای این

2 متد استفاده میشود , `future<Either<Failures, NumberTrivia>>` میباشد در اینجا قابلیت asynchronous بودن را اضافه میکند failures برای error handling استفاده میشود و `NumberTrivia` دیتایی است که در صورت success برگردانده میشود که در زیر کد قسمت توضیح داده شده را مشاهده میکنید.



The screenshot shows the Visual Studio Code editor with the file `number_trivia_repository.dart` open. The Explorer sidebar on the left shows the project structure, including the `lib` directory and its subdirectories. The main editor area displays the following Dart code:

```
lib > features > number_trivia > domain > repositories > number_trivia_repository.dart > ...  
1 import 'package:dartz/dartz.dart';  
2 import 'package:number_trivia/core/errors/failures.dart';  
3 import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';  
4  
5 abstract class NumberTriviaRepository {  
6   Future<Either<Failures, NumberTrivia>> getConcreteNumberTrivia(int number);  
7   Future<Either<Failures, NumberTrivia>> getRandomNumberTrivia(int number);  
8 }  
9
```


• Asynchronous

فرض کنید 2 function داریم که یکی asynchronous و دیگری synchronous میباشد در موقع اجرا کردن فایل منتظر نمیمانیم تا اجرای asynchronous function پایان یابد بلکه همزمان با اینکه آن اجرا میشود بقیه ی کد نیز اجرا میشود.

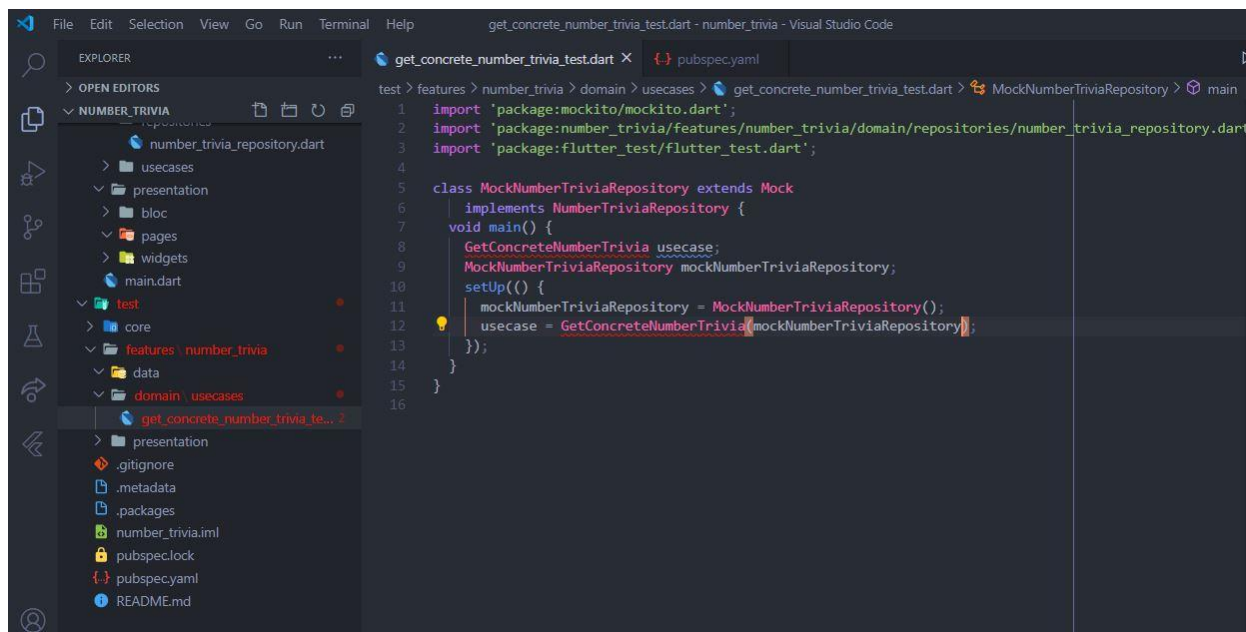
• نوشتن تست برای متد

getConcreteNumberTrivia()

تمامی تست ها در فولدر tests قرار میگیرد و معماری فولدرهای آن مانند معماری فولدر ها در lib میباشد نام فایل های تستی مانند فایل های production آنها میباشد منتهی در آخر آنها _test نیز قرار میگیرد. برای اضافه کردن functionality به تستمان NumberTriviaRepository را mock میکنیم و همچنین از این طریق میتوانیم بفهمیم آیا متدی کال شده است یا خیر.

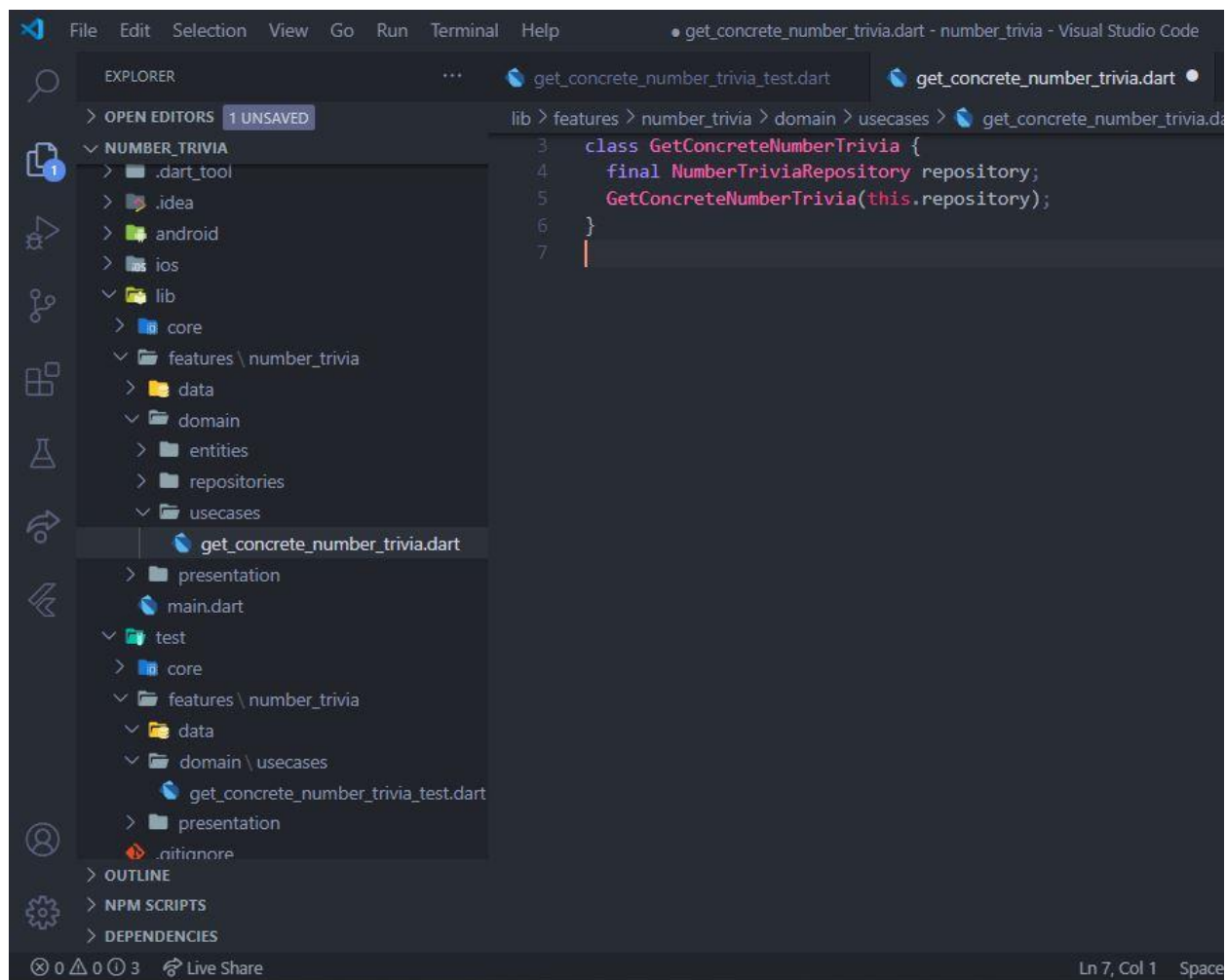
GetConcreteNumberTrivia نمونه ای از NumberTriviaRepository را از طریق constructor میگیرد. تست ها در دارت متدی به نام setUp() دارند که

این متد قبل از تست ها اجرا میشود در اینجا به آبجکتهایمان مقدار اولیه خواهیم داد همچنین تمامی تست ها در داخل `void main(){} اجرا میشوند` که در زیر مشاهده میکنید :



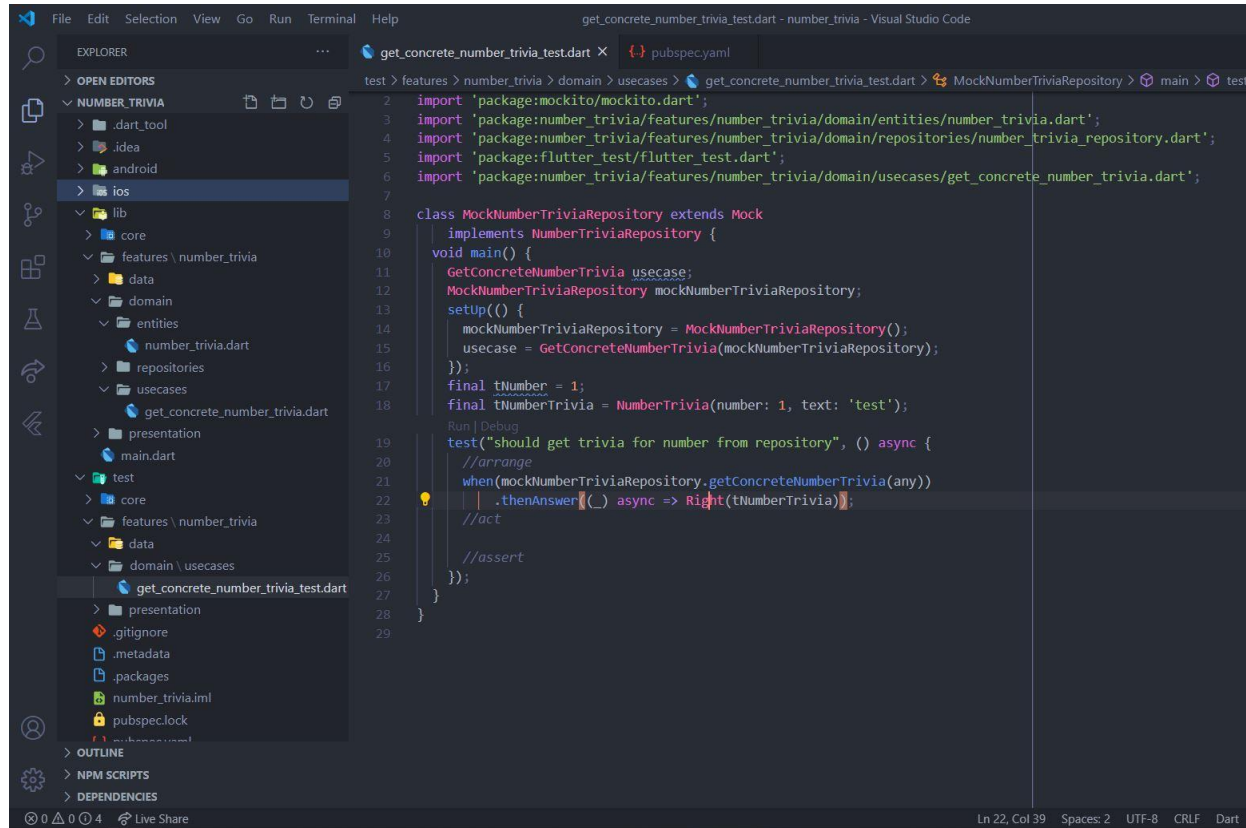
```
1 import 'package:mockito/mockito.dart';
2 import 'package:number_trivia/features/number_trivia/domain/repositories/number_trivia_repository.dart';
3 import 'package:flutter_test/flutter_test.dart';
4
5 class MockNumberTriviaRepository extends Mock
6   implements NumberTriviaRepository {
7   void main() {
8     GetConcreteNumberTrivia usecase;
9     MockNumberTriviaRepository mockNumberTriviaRepository;
10    setUp(() {
11      mockNumberTriviaRepository = MockNumberTriviaRepository();
12      usecase = GetConcreteNumberTrivia(mockNumberTriviaRepository);
13    });
14  }
15 }
16
```

همانطور که مشاهده میشود اروری وجود دارد زیرا کلاس `GetConcreteNumberTrivia` را تعریف نکرده ایم پس برای اینکه از حالت قرمز به حالت سبز تغییر وضعیت دهیم این فایل را درست میکنیم.



برای نوشتن بقیه ی تست باید بدانیم که در اینجا تستی که مینویسیم برای این است که مطمئن شویم که repository صدا زده شده و داده ها دست نخورده به usecase پاس داده میشوند. تست ما از 3 بخش assert , act , arrange تشکیل شده است که در بخش arrange هر موقع که متد getConcereteNumberTrivia() در کلاس MockNumberTriviaRepository با هر آرگومانی صدا

زده میشود همیشه با قسمت SUCCESS برای متد
getConcreteNumberTrivia() پاسخگو خواهیم بود.



```
2 import 'package:mockito/mockito.dart';
3 import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';
4 import 'package:number_trivia/features/number_trivia/domain/repositories/number_trivia_repository.dart';
5 import 'package:flutter_test/flutter_test.dart';
6 import 'package:number_trivia/features/number_trivia/domain/usecases/get_concrete_number_trivia.dart';
7
8 class MockNumberTriviaRepository extends Mock
9   implements NumberTriviaRepository {
10   void main() {
11     GetConcreteNumberTrivia usecase;
12     MockNumberTriviaRepository mockNumberTriviaRepository;
13     setUp(() {
14       mockNumberTriviaRepository = MockNumberTriviaRepository();
15       usecase = GetConcreteNumberTrivia(mockNumberTriviaRepository);
16     });
17     final tNumber = 1;
18     final tNumberTrivia = NumberTrivia(number: 1, text: 'test');
19
20     test("should get trivia for number from repository", () async {
21       //arrange
22       when(mockNumberTriviaRepository.getConcreteNumberTrivia(any))
23         .thenAnswer((_) async => Right(tNumberTrivia));
24       //act
25
26       //assert
27     });
28   }
29 }
```

در عکس زیر

```
expect(Right(tNumberTrivia), result);
```

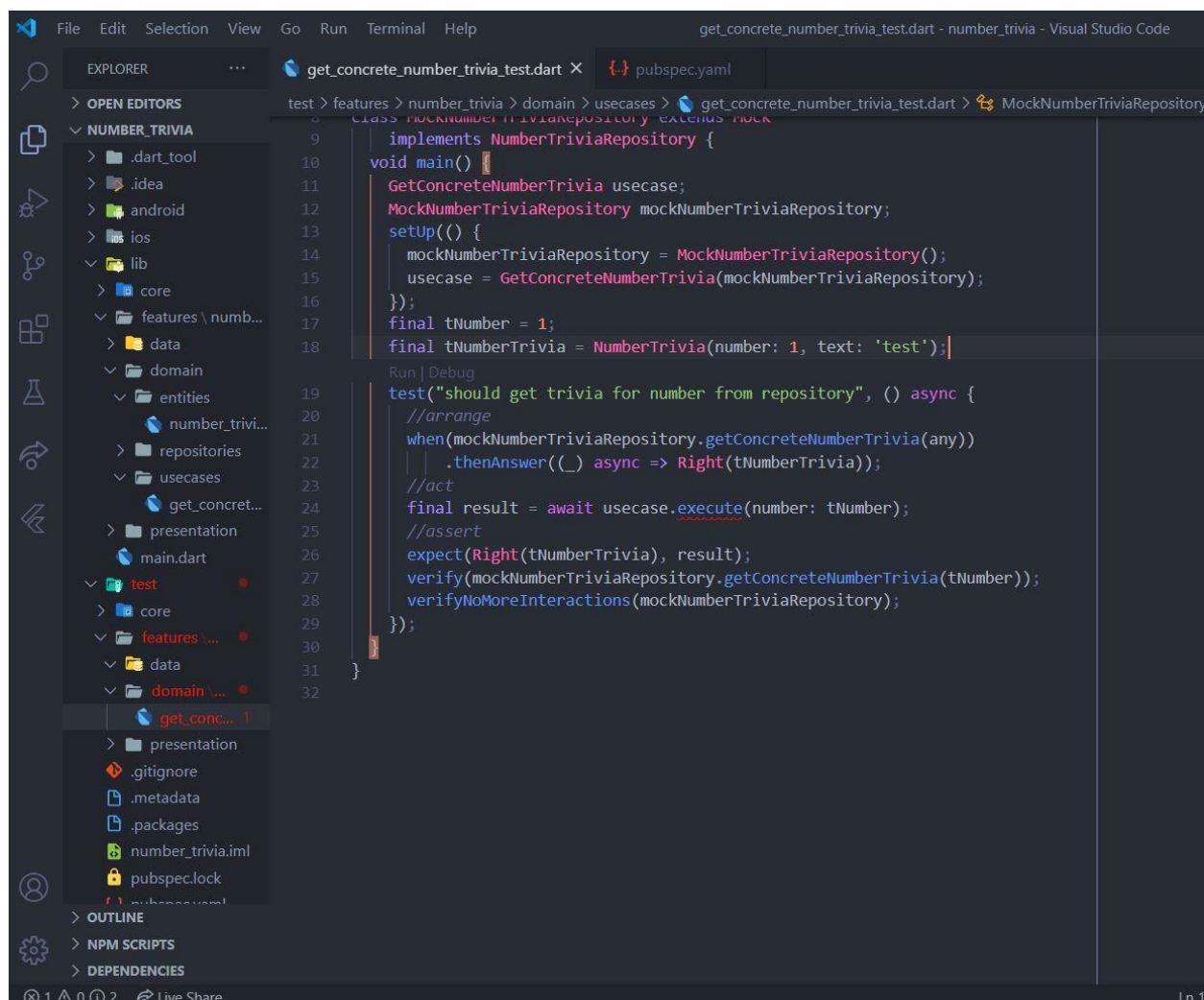
یعنی هر چیزی که repository برمیگرداند usecase نیز
باید برگرداند.

```
verify(mockNumberTriviaRepository.getConcreteNumberTrivia(tNumber));
```

در این قسمت مطمون میشویم که repository ما کال شده است.

`verifyNoMoreInteractions(mockNumberTriviaRepository);`

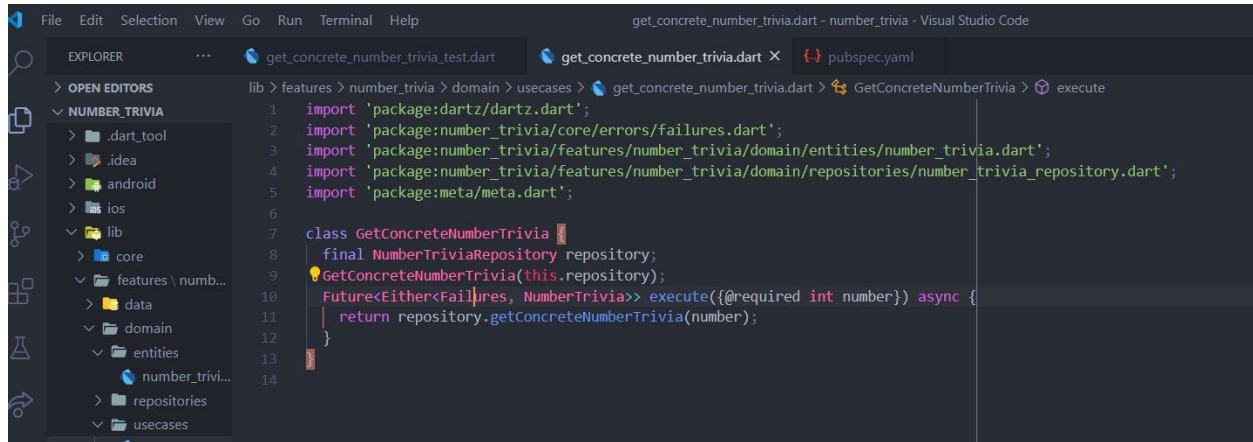
یعنی فقط متد بالایی باید کال شده باشد نه چیزی بیشتر.
که کد نهایی این بخشی به شکل زیر میباشد:



```
class MockNumberTriviaRepository extends Mock implements NumberTriviaRepository {
  void main() {
    GetConcreteNumberTrivia usecase;
    MockNumberTriviaRepository mockNumberTriviaRepository;
    setUp(() {
      mockNumberTriviaRepository = MockNumberTriviaRepository();
      usecase = GetConcreteNumberTrivia(mockNumberTriviaRepository);
    });
    final tNumber = 1;
    final tNumberTrivia = NumberTrivia(number: 1, text: 'test');

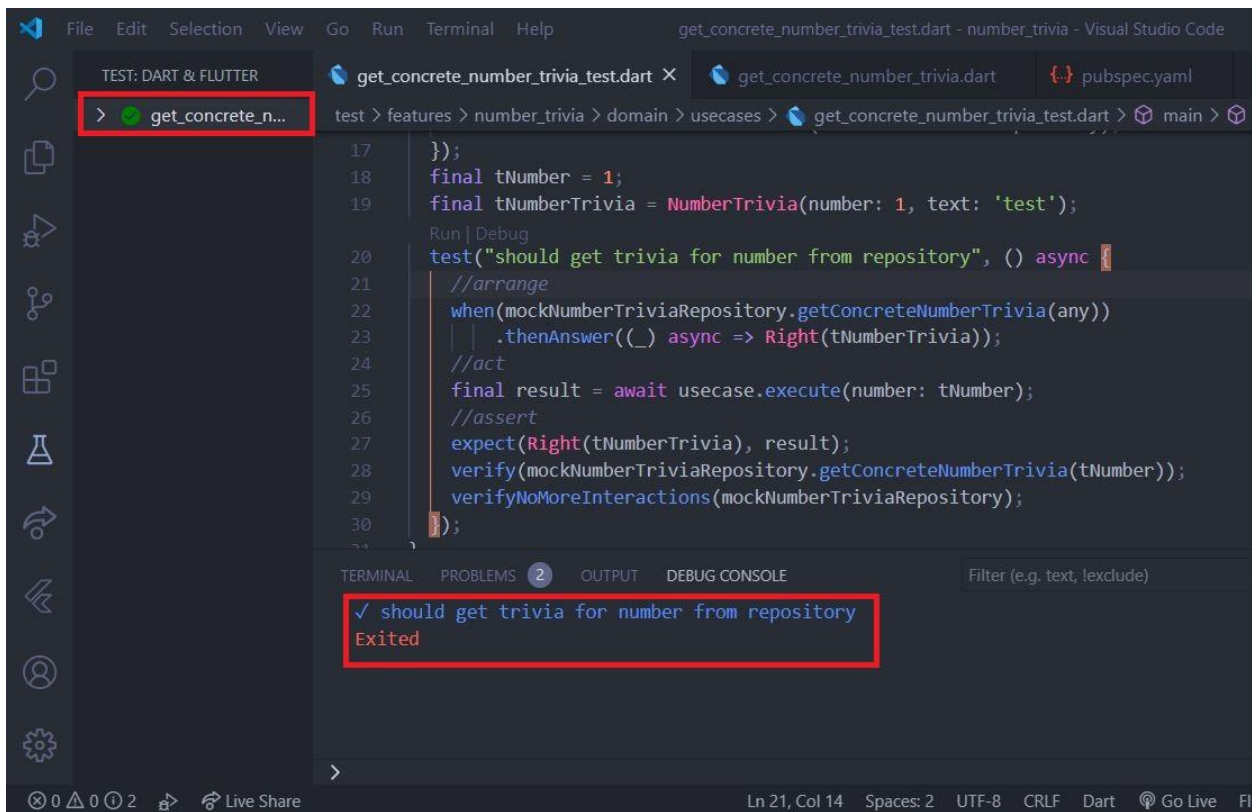
    test("should get trivia for number from repository", () async {
      //arrange
      when(mockNumberTriviaRepository.getConcreteNumberTrivia(any))
        .thenReturn((_) async => Right(tNumberTrivia));
      //act
      final result = await usecase.execute(number: tNumber);
      //assert
      expect(Right(tNumberTrivia), result);
      verify(mockNumberTriviaRepository.getConcreteNumberTrivia(tNumber));
      verifyNoMoreInteractions(mockNumberTriviaRepository);
    });
  }
}
```

برای اینکه ارور بالا را از بین ببریم متد execute را به usecase مان اضافه خواهیم کرد :



• نحوه ی ران کردن تست

ابتدا به قسمت `file > preferences > keyboard shortcuts` میرویم سپس `dart:run all tests` را انتخاب کرده سپس shortcut یی را برایش تعیین میکنیم. بعد از مشخص شدن آن تست را اجرا خواهیم کرد.



قسمت های نشان داده شده در عکس بالا نشان دهنده ی این است که تست ما با موفقیت انجام شده است.

• Callable classes

در دارت متدی به نام call داریم که میتوانیم از طریق object.call() و یا object() صدا کنیم پس میتوانیم در کلاس GetConcreteNumberTrivia بجای استفاده از execute از call استفاده کنیم که باعث جلوگیری از ریداندنسی میشود.

در فایل `get_concrete_number_trivia_test.dart` داریم :

```
final result = await usecase(number: tNumber);
```

در فایل `get_concrete_number_trivia.dart` داریم :

```
Future<Either<Failures, NumberTrivia>> call({  
  @required int number}) async {  
  return repository.getConcreteNumberTrivia(number);  
}
```

Endpoint ها در 2 usecase متفاوت میباشد یعنی دیتا ها برای آنها از آدرس های متفاوتی میایند. برای یکدست کردن usecase هایمان و همچنین برای جلوگیری از اینکه یک usecase call را صدا کند و دیگری execute را صدا کند از یک abstract کلاسی استفاده میکنیم که تمام usecase ها از آن نشأت بگیرند. چون این کلاس قرار است بین خیلی

از فایل ها استفاده شود پس آنرا در فولدر core قرار میدهیم. و همچنین ما کلاس های abstract را تست نمیکنیم. در جاوا کاتلین و ... interface داریم و لی در دارت نداریم و بجایش از abstract class ها استفاده میکنیم. در کد زیر همانطور که در GetConcreteNumberTrivia() نمایش داده شد call از نوع <<Future<Either<Failures, Type>>> میباشد. چون ممکن است return type برای همه ی usecase ها NumberTrivia نباشد پس از Type استفاده میکنیم و همچنین params نیز نشان دهنده ی تمام پارامتر های متد call میباشد چون ممکن است کلاسی پارامتری را نگیرد در زیر کلاس NoParams() را نیز اضافه میکنیم.

```
import 'package:dartz/dartz.dart';
import 'package:number_trivia/core/errors/failures.dart';

abstract class UseCase<Type, Params> {
  Future<Either<Failures, Type>> call(Params params);
}
```

```
}  
  
class NoParams {}
```

در زیر کلاس () GetConcreteNumberTrivia که از usecase نشأت گرفته است را مشاهده میکنید. کلاس params بصورت جدا برای تمیز تر شدن کد ساخته شده است.

```
import 'package:dartz/dartz.dart';  
import 'package:equatable/equatable.dart';  
import 'package:number_trivia/core/errors/failures.dart';
```

```
import 'package:number_trivia/core/usecases/usecase.dart';
import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';
import 'package:number_trivia/features/number_trivia/domain/repositories/number_trivia_repository.dart';
import 'package:meta/meta.dart';

class GetConcreteNumberTrivia implements UseCase<NumberTrivia, Params> {
  final NumberTriviaRepository repository;
  GetConcreteNumberTrivia(this.repository);
  @override
  Future<Either<Failures, NumberTrivia>> call(Params params) async {
    return repository.getConcreteNumberTrivia(params.number);
  }
}
```

```
class Params {
    final int number;

    Params({@required this.number});
}
```

بعد از این مرحله مشاهده میکنیم که فایل تست این usecase دچار خطا شده است پس بصورت زیر خطا را اصلاح میکنیم

```
final result = await usecase(Params(number: t
Number));
```

• کلاس GetRandomNumberTrivia

ابتدا شروع به نوشتن تست میکنیم که بسیار شبیه به تست GetConcreteNumberTrivia() میباشد در این usecase عدد در api مان generate میشود پس

آرگومانی برای گرفتن نداریم پس در این تست tNumber را حذف میکنیم که به شکل زیر است :

```
import 'package:dartz/dartz.dart';
import 'package:mockito/mockito.dart';
import 'package:number_trivia/core/usecases/usecase.dart';
import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';
import 'package:number_trivia/features/number_trivia/domain/repositories/number_trivia_repository.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:number_trivia/features/number_trivia/domain/usecases/get_random_number_trivia.dart';
```

```
class MockNumberTriviaRepository extends Mock
    implements NumberTriviaRepository {}

void main() {
    MockNumberTriviaRepository mockNumberTriviaRepository;
    GetRandomNumberTrivia usecase;
    setUp(() {
        mockNumberTriviaRepository = MockNumberTriviaRepository();
        usecase = GetRandomNumberTrivia(mockNumberTriviaRepository);
    });
    final tNumberTrivia = NumberTrivia(number: 1, text: "test");
    test('should get trivia from the repository', () async {
        when(mockNumberTriviaRepository.getRandomNumberTrivia())
```

```

        .thenAnswer((_) async => Right(tNumberTr
ivia));
    final result = await usecase(NoParams());
    expect(result, Right(tNumberTrivia));
    verify(mockNumberTriviaRepository.getRandomNumberTrivia());
    verifyNoMoreInteractions(mockNumberTriviaRepository);
  });
}

```

این فایل دارای ارور میباشد زیرا هنوز usecase مان را
تعریف نکرده ایم که در زیر اینکار را میکنیم :

```

import 'package:number_trivia/core/errors/failur
es.dart';
import 'package:dartz/dartz.dart';

```

```
import 'package:number_trivia/core/usecases/usecase.dart';
import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';
import 'package:number_trivia/features/number_trivia/domain/repositories/number_trivia_repository.dart';

class GetRandomNumberTrivia implements UseCase<NumberTrivia, NoParams> {
  final NumberTriviaRepository repository;
  GetRandomNumberTrivia(this.repository);
  @override
  Future<Either<Failures, NumberTrivia>> call(NoParams params) async {
    return await repository.getRandomNumberTrivia();
  }
}
```


در صورتی که تست را اجرا کنیم متوجه میشویم که تست با موفقیت انجام میشود.

• Json

کوتاه شده ی عبارت JavaScript Object Notation میباشد که فرمتی کم حجم برای ذخیره سازی و انتقال داده هاست معمولا برای انتقال داده بین سرور و صفحهای وب از این فرمت استفاده میشود که برای اینکار نوع دیتا تنها میتواند تکست باشد.

• Serialization

انکود یا سریالایز کردن یعنی دیتا استراکچرمونو به string تبدیل کنیم که برای این کار 2 روش داریم :

1. روش دستی

2. روش auto generated

روش اول برای پروژه های کوچک که مدل های کمی دارند استفاده میشود و روش دوم برای پروژه های خیلی بزرگ با مدل های بسیار زیاد بکار میرود.

Deserialization •

دیکود یا دیسریالایز کردن یعنی string را به دیتا استراکچر تبدیل کنیم.

با استفاده از پکیج **dart:convert** میتوانیم فایل های json را دیکود کرده و در فلاتر به فرمت `Map<String, dynamic>` تبدیل کنیم

json زیر برای کاربر را در نظر بگیرید :

```
{  
  "name": "John Smith",  
  "email": "john@example.com"  
}
```

برای ساختن مدل کاربر طبق زیر عمل میکنیم :

کاری که مدل زیر اینجا میدهد این است که در ابتدا 2 پارامتر name , email را میپذیرد سپس با استفاده از دستور `fromJson` آنرا به فرمتی که دارت آنرا میپذیرد در میآوریم

در اینجا `json['name']` میشود همان `'john smith'` و
`json['email']` میشود `'john@example.com'`

```
class User {  
  final String name;  
  final String email;  
  
  User(this.name, this.email);  
  
  User.fromJson(Map<String, dynamic> json)  
    : name = json['name'],  
      email = json['email'];  
  
  Map<String, dynamic> toJson() =>  
  {  
    'name': name,  
    'email': email,  
  };  
}
```

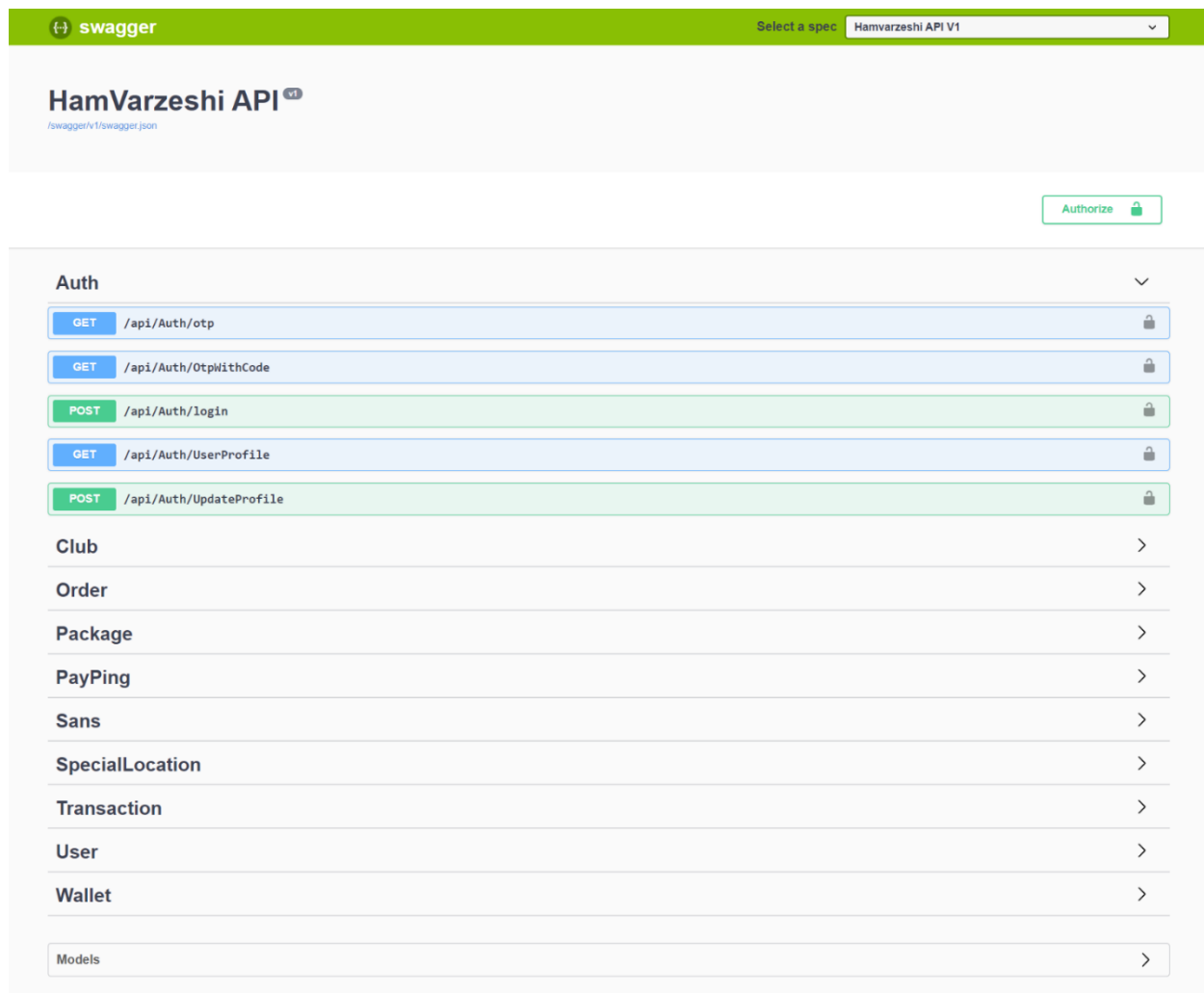
• swagger

در swagger میتوانیم ساختار api ها را تعریف کنیم به عنوان مثال پروژه ی همورزشی را در نظر بگیرید این پروژه برای کاربرانی تعریف شده که بدنبال باشگاه و ورزش کردن در آنها میباشند.

این پروژه از ده ها api تشکیل شده است که به قسمتهایی نظیر

auth,club,order,package,payping,sans,specialL
ocation,Transaction,User,Wallet تقسیم شده است

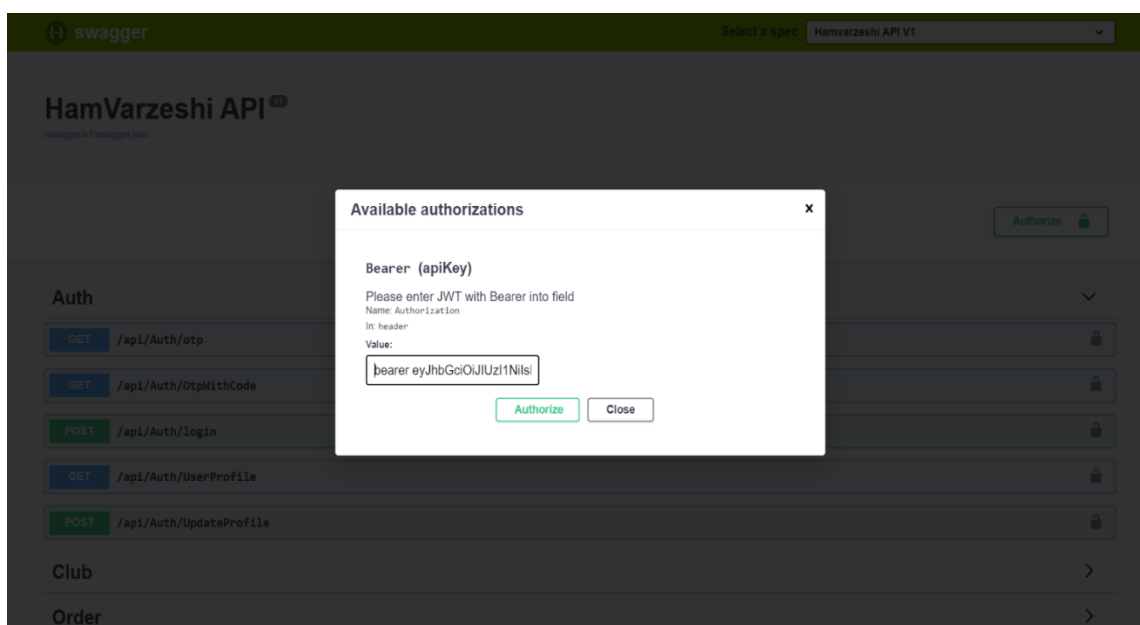
که در هر کدام از این بخش ها ده ها api وجود دارد. که هر کدام از این Api ها از GET و یا POST استفاده میکنند از متد GET برای گرفتن داده ها از سرور و از POST برای فرستادن دیتا به سرور استفاده میشود. محیطی از swagger را در شکل زیر میتوانید مشاهده کنید



در این پروژه از ترکیب postman و swagger استفاده میکنیم دلیل آن هم این است که در swagger هر دفعه که پیج ریفرش میشود و یا این تب را دوباره باز میکنیم نیاز است که دوباره authorize کنیم ولی در postman این کار تنها یک بار برای همیشه کافیست.

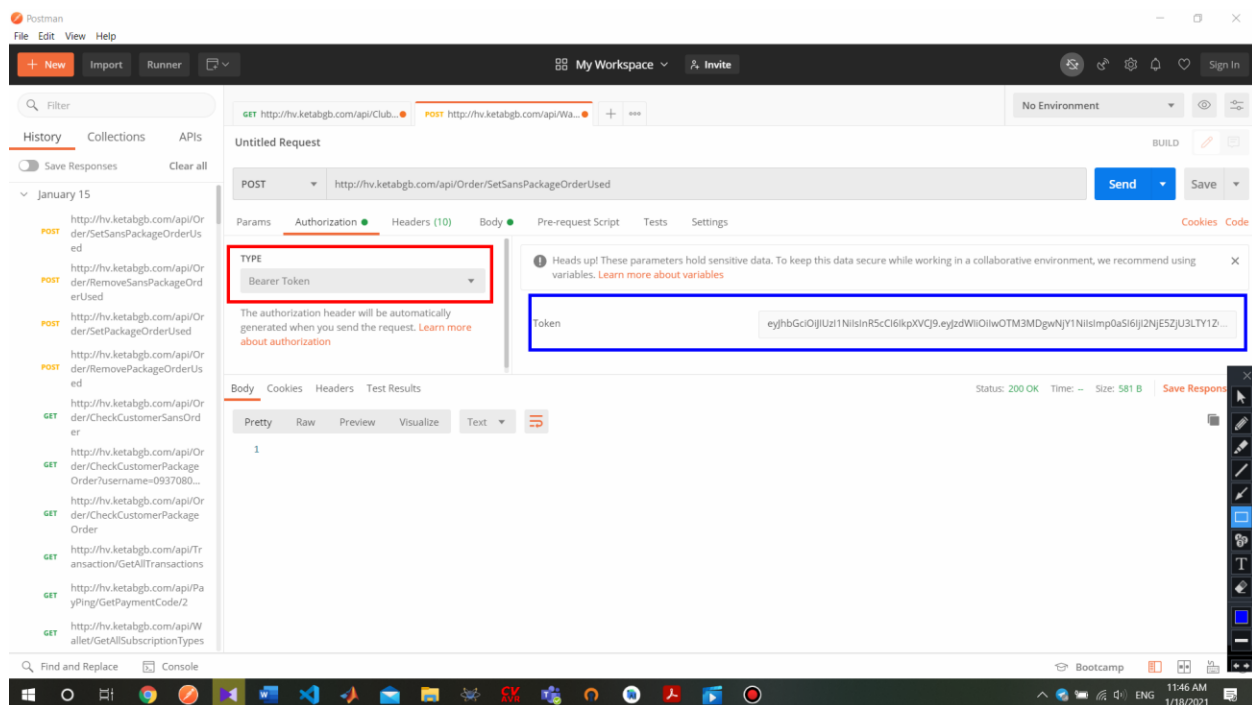
- نحوه ی authorize کردن در swagger

1. بر روی گزینه ی authorize بالا سمت راست صفحه کلیک میکنیم
2. برای authorize کردن دستور 'bearer token' را میزنیم که در این صورت authorize میشود.

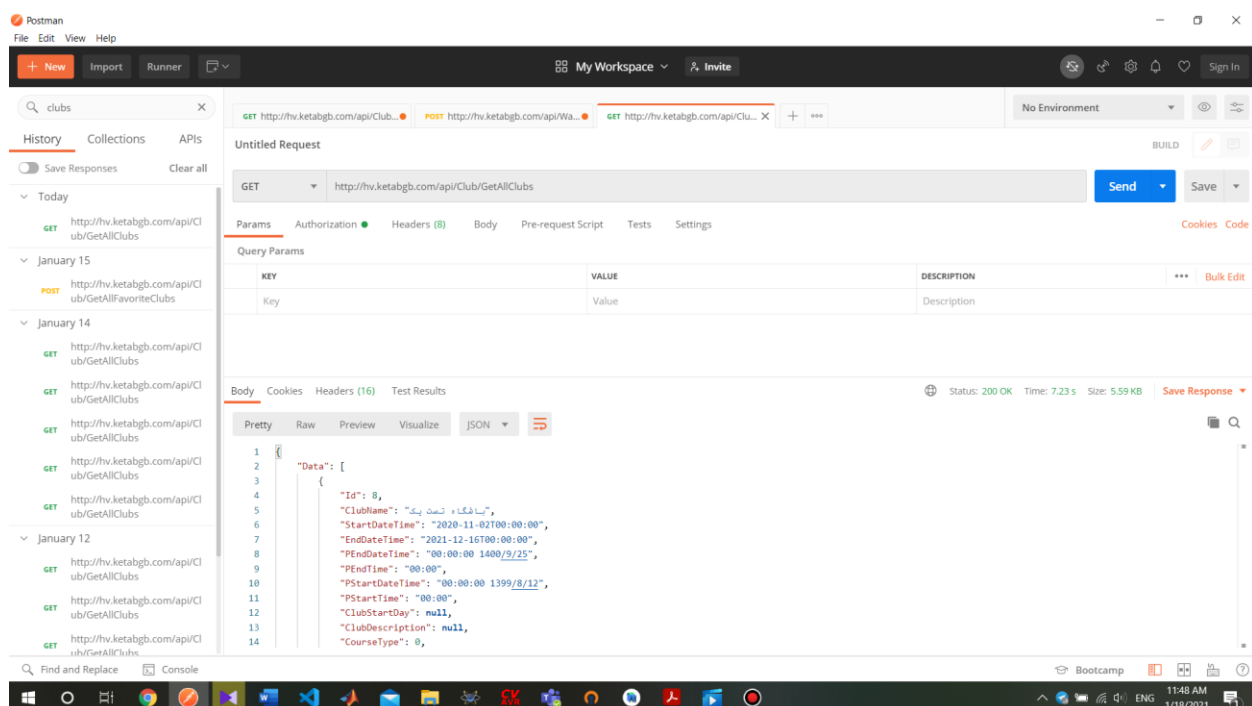


- نحوه ی authorize کردن در postman

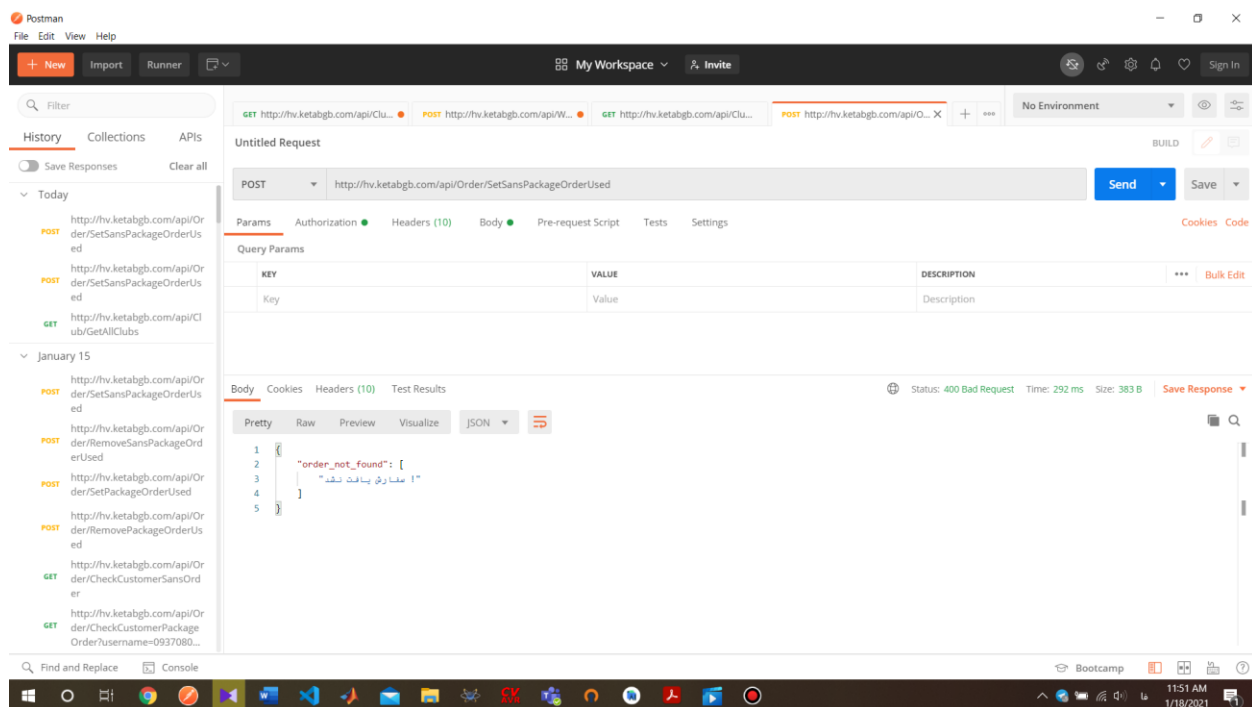
ابتدا وارد تب authorization میشود سپس طبق کادر قرمز رنگ type را بر روی bearer token قرار میدهیم و در قسمت آبی رنگ مقدار token را وارد میکنیم.



برای استفاده از GET ابتدا url برای api مورد نظر را وارد میکنیم و نوع متد را بر روی GET قرار میدهیم سپس بر روی گزینه send کلیک کرده که نتیجه ی زیر بدست میاید :



برای استفاده از متد POST نیز url مورد نظر را انتخاب کرده و نوع متد را بر روی POST قرار میدهیم که در این صورتی حاصلی مانند زیر بدست میاید

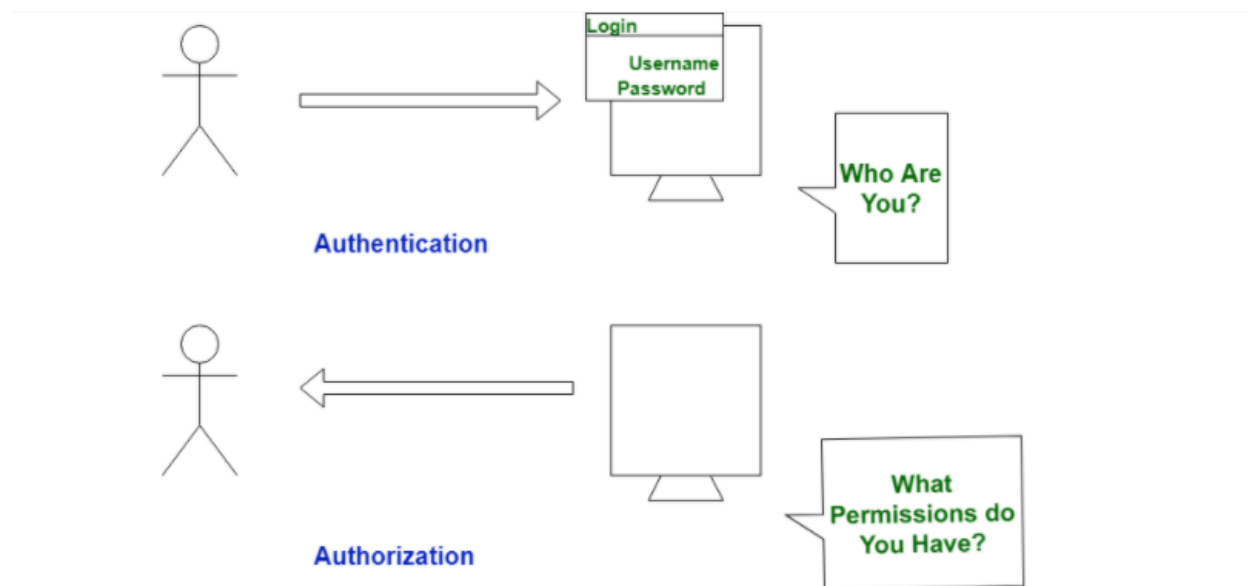


یادمان باشد برای دسترسی به کل api ها نیاز داریم که authorize شده باشیم

• فرق authentication و authorization

هر دوی آنها برای امنیت سیستم ما بکار میروند. در کل authentication قبل از authorization بکار میرود در

authentication ابتدا مشخص میشود که شما که هستید و در صورتی که توسط سیستم شناسایی شدید سپس میتوانید از قابلیت های اپلیکیشن استفاده کنید در مرحله بعدی authorization است که وقتی در ابتدا کاربر ما authenticate شد سیستم بررسی میکند ببیند این فرد در مرحله ی بعدی در ورود به سیستم باید به چه نوع دسترسی ای دسترسی داشته باشد آیا کاربر این سیستم است ؟ اگر هست مثلا مدیر باشگاه هست یا مشتری؟



• نحوه ی autogenerate کردن

ابتدا وارد swagger شده سپس وارد قسمت api های اپلیکیشن میشویم که به شکل زیر است :

```

{
  swagger: "2.0",
  - info: {
    version: "v1",
    title: "HamVarzeshi API"
  },
  - paths: {
    - /api/Auth/otp: {
      - get: {
        - tags: [
          "Auth"
        ],
        operationId: "SendOtp",
        consumes: [ ],
        - produces: [
          "text/plain",
          "application/json",
          "text/json"
        ],
        - parameters: [
          - {
            name: "username",
            in: "query",
            required: false,
            type: "string"
          }
        ],
        - responses: {
          - 200: {
            description: "Success"
          },
          - 400: {
            description: "Bad Request",
            - schema: {
              $ref: "#/definitions/ProblemDetails"
            }
          }
        }
      }
    },
    - /api/Auth/OtpWithCode: {
      - get: {

```

• لایه data

لایه ی دیتا با Api ها و کتابخانه ها سر و کار دارد. طبق شکل تمام لایه های بیرونی به لایه های درونی وابسته هستند پس منطقیه که از entities شروع کنیم زیرا در صورتی که entities را نداشته باشیم usecase ها ما چه چیزی را میخواستند return کنند طبق این صحبت برای لایه ی دیتا نیز با توجه به شکل اول باید models را درست کنیم که کارش تبدیل fromjson و یا tojson میباشد زیرا

در نهایت از api هایمان json میگیریم که باید در نهایت به فرمت قابل خواندن برای dart تبدیل شود در واقع model ها همان entity ها هستند که قابلیت from/to json به آنها اضافه شده است. Contract ها که به صورت abstract class تعریف میشود به ما اجازه میدهند که از داخلی ترین قسمت به خارجی ترین قسمت ها توسعه ی خود را انجام بدهیم. هنگامی که از logic استفاده میکنیم یعنی باید توسعه ی ما از روش TDD باشد

• مدل ها (models)

همانطور که گفته شد در این قسمت باید مدلی را بنویسیم که subclass برای NumberTrivia محسوب میشود پس ابتدا تست انرا مانند زیر مینویسیم :

```
import 'package:flutter_test/flutter_test.dart';
import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';

void main() {
  final tNumberTriviaModel = NumberTriviaModel
(number: 1, text: "test");
```

```
test('number trivia model is subclass of number
trivia', () async {
  expect(tNumberTriviaModel, isA<NumberTrivi
a>());
});
}
```

در مرحله بعدی کلاس number_trivia_model.dart را
پایاده سازی میکنیم که به شکل زیر میباشد:

```
import 'package:flutter/cupertino.dart';
import 'package:number_trivia/features/number
_trivia/domain/entities/number_trivia.dart';

class NumberTriviaModel extends NumberTrivia
{
  NumberTriviaModel({@required String text, @r
equired int number})
    : super(text: text, number: number);
```

```
}
```

در ابتدا متد fromJson را پیاده سازی میکنیم که نمونه ای از NumberTriviaModel را return میکند که دیتای داخل آن همان دیتای داخل string میباشد. Json ها را از طریق fixture میگیریم که فایل json است که برای تست کردن بصورت مدام بکار میرود. response ما از طریق random endpoint بصورت زیر میباشد :

<http://numbersapi.com/random/trivia?json>

عدد ها لزوما همیشه integer نیستند و میتوانند double نیز باشند.

"number": 4e+185 در دارت این عدد int محسوب نمیشود بلکه double میباشد. Fixtures در داخل فولدر test قرار میگیرند دیتا ها در داخل fixtures باید دارای تمامی فیلد ها باشند. دیتا ها در داخل fixtures میتوانند متفاوت باشند اما structure کلی ریسپانس واقعی با

ریسپانس در داخل fixtures باید یکی باشند. در زیر فایل trivia.json مشاهده میشود :

```
{  
  "text": "Test code",  
  "number": 1,  
  "found": true,  
  "type": "trivia"  
}
```

در زیر فایل trivia_double.json را مشاهده میکنید.

```
{  
  "text": "Test code",  
  "number": 1.0,  
  "found": true,  
  "type": "trivia"  
}
```

فایل های تست ما نمیتوانند با قایل ها مستقیم کار کنند پس باید به آنها String پاس بدهیم که برای اینکار ابتدا

فایلی به نام `fixture_reader.dart` در همین فولدر فعلی تشکیل داده و سپس آدرس فایل را بصورت `string` میخوانیم.

```
import 'dart:io';

String fixture(String name) => File('test/fixtures/
$name').readAsStringSync();
```

حال نوبت تست کردن `trivia.json` رسیده است برای اینکار باید از دیتاهای `trivia.json` را از طریق `fixture` دریافت کنیم و سپس `json` را دیکود کرده و به فرم `Map<String, dynamic>` دربیابیم :

```
group('return a valid model when json number i
s an integer', () async {
  //arrange
  final Map<String, dynamic> mapJson =
    json.decode(fixture('trivia.json'));
```

```
//act  
//assert  
});
```

در ادامه mapJson را به NumberTriviaModel پاس
میدهیم و انتظار داریم tNumberTriviaModel = result
باشد :

```
group('return a valid model when json number i  
s an integer', () async {  
  //arrange  
  final Map<String, dynamic> mapJson = json.d  
ecode(fixture('trivia.json'));  
  //act  
  final result = NumberTriviaModel.fromJson(m  
apJson);  
  //assert  
  expect(tNumberTriviaModel, result);  
});
```


در فایل `number_trivia_model.dart` باید فانکشن `fromJson` تعریف شود که از نوع `factory` است.

• Factory

نوعی فانکشن یا متد میباشد که `object` هایی از انواع مختلف را برمیگرداند.

```
number: (json['number'] as num).toInt(),
```

`num` در بالا شامل هم `int` و هم `double` میشود که در نهایت جفتشان به `int` تبدیل خواهند شد.

```
import 'package:meta/meta.dart';
import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';

class NumberTriviaModel extends NumberTrivia
{
  NumberTriviaModel({
    @required String text,
    @required int number,
  }) : super(text: text, number: number);
```

```
factory NumberTriviaModel.fromJson(Map<String, dynamic> json) {
  return NumberTriviaModel(
    text: json['text'],
    number: (json['number'] as num).toInt(),
  );
}
```

در قسمت بعدیتست را برای toJson مینویسیم که فقط فیلدهای text, number را نیاز داریم زیرا فقط این 2 فیلد در داخل number_trivia موجو میباشند :

```
group('toJson', () {
  test('should return a json map', () async {
    final expectedMap = {
      "text": "Test code",
      "number": 1,
    };
    final result = tNumberTriviaModel.toJson();
```

```
expect(result, expectedMap);  
});  
});
```

در نهایت نیز فایل مدل را به شکل زیر کامل میکنیم :

```
import 'package:meta/meta.dart';  
import 'package:number_trivia/features/number_trivia/domain/entities/number_trivia.dart';  
  
class NumberTriviaModel extends NumberTrivia  
{  
  NumberTriviaModel({  
    @required String text,  
    @required int number,  
  }) : super(text: text, number: number);  
  
  factory NumberTriviaModel.fromJson(Map<String, dynamic> json) {  
    return NumberTriviaModel(  
      text: json['text'],
```

```
number: (json['number'] as num).toInt(),
);
}

Map<String, dynamic> toJson() {
  return {
    'text': text,
    'number': number,
  };
}
}
```

Repository مغز لایه ی دیتا میباشد زیرا این قسمت وظیفه ی رسیدگی به داده ها ، کش کردن و تصمیم گیری برای اینکه از remote و یا local data source استفاده کنیم میگیرد. ساختار سطحی لایه ی دیتا را در بالا نشان دادیم در زیر می خواهیم پیاده سازی در لایه ی دیتا را انجام دهیم.

همانطور که قبل نیز اشاره شد contract وابستگی را از بین میبرد.

تعریف آن مانند زیر میباشد :

```
class NumberTriviaRepositoryImpl implements
NumberTriviaRepository {
    @override
    Future<Either<Failures, NumberTrivia>> getConcreteNumberTrivia(int number) {
        // TODO: implement getConcreteNumberTrivia
        throw UnimplementedError();
    }

    @override
    Future<Either<Failures, NumberTrivia>> getRandomNumberTrivia() {
        // TODO: implement getRandomNumberTrivia
        throw UnimplementedError();
    }
}
```

همانطور که برای repository ابتدا contract نوشتیم و سپس در قسمت تست آنرا mock کردیم همین کار را برای data source هایمان نیز انجام میدهیم. که این تنها کافی نیست و باید راهی را پیدا کنیم که بفهمیم کاربر ما به اینترنت متصل است یا خیر زیرا در صورتی که متصل نباشد اطلاعات و دیتاها را از cache کاربر میگیریم. برای اینکار فایلی به نام network_info.dart را در فولدر core داخل فولدر platform ایجاد میکنیم که دلیل ساخت این فایل در فولدر platform این است که چک کردن وضعیت اینترنت برای اندروید و iOS ممکن است یکسان نباشد

```
abstract class NetworkInfo {  
    Future<bool> get isConnected;  
}
```

• Remote data source

شبيه به repository پیاده سازی میشود فقط return type آن متفاوت میباشد و مدل برمیگرداند که دیتای آن به

فرمت قابل خواندن برای فلاتر تبدیل میشود برای مدیریت ارور ها نیز از exception استفاده میکنیم.

در صورتی که مثلا ارور کد 404 که یعنی آیتم مورد نظر پیدا نشد داشته باشیم. در این متد ها باید exception داشته باشیم که بتوانیم این ارور ها را handle کنیم.

```
abstract class NumberTriviaRemoteDataSource {  
    Future<NumberTriviaModel> getConcreteNumberTrivia(int number);  
    Future<NumberTriviaModel> getRandomNumberTrivia();  
}
```

برای handle کردن exception ها فایلی به نام exceptions را در داخل فولدر ارور ها ایجاد میکنیم در این فایل کلاسی برای ارور ها سمت سرور و کلاسی را برای ارور های کش تعریف میکنیم. این exception ها در ابتدا توسط repository گرفته شده و به failure تبدیل میشود و سپس از طریق either برگردانده میشوند.

```
class ServerException implements Exception {}  
class CacheException implements Exception {}
```

همانطور که میدانیم failure ها در داخل فایل failures
اتفاق میفتند که برای این exception ها باید failure هم
تعریف کنیم که به شکل زیر می باشد :

```
abstract class Failures {  
    final List properties;  
  
    Failures(this.properties);  
}  
  
class ServerFailure extends Failures {  
    ServerFailure(List properties) : super(properties);  
}  
  
class CacheFailure extends Failures {  
    CacheFailure(List properties) : super(properties);  
}
```


Local Data Source •

تا به اینجا چه برای مدل ها و چه برای انتیتی ها دیت را گرفتیم ولی در این قسمت می‌خواهیم هم دیتا را وارد کش کرده و هم از آن بگیریم و برای ما مهم نیست که متدی که استفاده میکنیم random است یا remote زیرا آخرین باری که کاربر به اینترنت وصل باشد و با استفاده از یکی از این متد ها trivia را بگیرد باید آن دیتا را در داخل کش ذخیره کنیم و در صورتی که کاربر به اینترنت وصل نبود داده را از کش به کاربر نمایش می‌دهیم.

```
import 'package:number_trivia/features/number_trivia/data/model/number_trivia_model.dart';

abstract class NumberTriviaLocalDataSource {
    Future<NumberTriviaModel> getLastNumberTrivia();
}
```

```
Future<void> cachNumberTrivia(Number  
TriviaModel triviaToCache);  
}
```

در صورتی که کاربر به اینترنت متصل نباشد و همچنین و داده ای هم کش نشده باشد exception یی به نام cacheException خواهیم داشت. دلیل اینکه دیتا سورس هایمان را در داخل repository تعریف نمیکنیم این است که می‌خواهیم آن‌ها را مستقل از سایر لایه‌ها و فایل‌هایی که تولید کردیم داشته باشیم و همچنین در متد cachNumberTrivia() داده یا همان json که گرفتیم را با استفاده از پکیج shared_preferences در داخل کش گوشی قرار می‌دهیم.

• متداول ترین ارورهای http

ممکن است این ارور ها از طرف کاربر و یا از طرف سرور باشد که اکثرا هم از طرف سرور می‌باشد.



هر اروری از نوع HTTP کد خاص به خود را دارد مثلاً ارور 404 یعنی خواستید به صفحه ای متصل بشوید که وجود خارجی ندارد.

• HTTP Error 401 (Unauthorized)

این ارور موقعی اتفاق میفتد که کاربر بخواهد به صفحه ای دسترسی داشته باشد که اجازه ی ورود به آنرا ندارد اغلب اوقات این اتفاق موقعی میفتد که کاربر موفق به لاگین کردن نشده است حتی در postman هم این ارور رایج است که توسعه دهنده قبل از ورود token بخواهد به صفحه ای دسترسی داشته باشد که در این صورت با چنین اروری مواجه خواهد شد.

• HTTP Error 400 (Bad Request)

یعنی کاربر اشتباه وارد شده و یا اینکه به شیوه ای request کاربر در هنگام رسیدن به سرور خراب شده است.

• HTTP Error 404 (Not Found)

این ارور که بسیار متداول بین کاربران است نیز یعنی کاربر میخواهد وارد صفحه ای شود و یا url یی را وارد کرده که یا خراب است یا وجود ندارد و وب مستر لینک را تغییر داده است.

• HTTP Error 500 (Internal Server Error)

طبق آمارهای گوگل این نوع ارور از 404 نیز متداول تر میباشد و همانطور که از اسمش پیداست یعنی مشکل از خود سرور میباشد و ممکن است سرور نتوانسته باشد overload شود و به request های کاربر نتواند رسیدگی کند.

• پیشنهادات

قبل از یادگیری این دسیپلین تمامی کدهای بنده اسپاگتی بودند و وقتی بعد از چندین ماه میخواستیم تغییری در این کدها ایجاد کنیم خیلی کار سخت میشد مخصوصا وقتی که شخص جدیدی درگیر کار با کدها بشود ولی بعد از یادگیری این دسیپلین و داکيومنت کردن هر مرحله ، تمامی api ها و داکيومنت کردن مشکلات و راه حل آنها هم برای بنده هم برای شخص دیگری که بعدا به کدها مراجعه میکند کار بسیار ساده و راحت شد.

• خلاصه کارنهایی

در فصل اول با مکان کارآموزی و افراد در این محل آشنا شدیم در فصل دوم به بررسی مفاهیم اولیه پرداختیم و در فصل سوم بیشتر به پیاده سازی و همچنان یادگیری بیشتر مفاهیم پرداخته شد.

• کارهای آتی

در آینده به پیاده سازی این دسیپلین بر روی اپلیکیشن هم ورزشی ، یادگیری graphql و ساده تر کردن و داکيومنت کردن تمامی کارها و ... میپردازم.

<http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd>

<https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

یادگیری کلی این دیسیپلین از طریق :

https://resocoder.com/?s=+flutter+tdd&tcb_sf_D=post&tcb_sf_post_type%5B%5D_post_type_D=page%5B%5D

[9%D9%85%D2%A8%https://www.roxo.ir/%D](#)

[-4B8%D2%B8%D88%](#)

[9%C%D%8DB1%B8%D2%B8%AA%D8%%D](#)

[-82](#)

[8%D3%B8%D8%A8%D7%A8%D9%88%%D](#)

[/C%8AA%DA%AF%DB](#)

فرم ها

فرم شماره 1

بسمه تعالی

دانشگاه آزاد اسلامی
واحد تهران مرکزی



فرم خلاصه اطلاعات کارآموزی

نام و نام خانوادگی دانشجو: <u>یلیم جعفری</u>	استاد کارآموزی: <u>ایمن محارر زاراد</u>
شماره دانشجویی: <u>960164039</u>	سرپرست کارآموزی: <u>امید خنسن امینی</u>
رشته / گرایش: <u>مهندسی کامپیوتر / نرم افزار</u>	ترم و مدت کارآموزی: <u>ترم پاییز سال 1399-1400</u> <u>کلاس کارآموزی: 1884156372</u>
مقطع تحصیلی: <u>کارشناسی پیوسته</u>	تاریخ و امضاء دانشجو: <u>1399 / 11 / 20</u>
نام محل کارآموزی: <u>شرکت نوآوران ورزشی و بهداشت</u>	آدرس و تلفن کارآموزی: <u>آدرس: خیابان آراک، خیابان ماسی، جنب چهار دانشگاهی، دانشگاه صنعتی شریف،</u> <u>پلاک 79، تلفن: 021-6691105</u>
سرپرست کارآموزی: <u>ابیر حسن امینی</u>	
عنوان و موضوع کارآموزی: <u>یادگیری و انجام پروژه به وسیله ی رایسپیلین TDD</u>	
شروع و روزهای کارآموزی: <u>شروع 1399، 11، 18</u>	
روزهای کارآموزی: <u>هر هفته و هر روز به مدت 6 ساعت</u>	
تاریخ و امضاء استاد کارآموزی:	

دانشگاه آزاد اسلامی

واحد تهران مرکزی



گزارش پیشرفت کارآموزی

نام و نام خانوادگی دانشجو: <u>سید مجتبی</u>	استاد کارآموزی: <u>اسدالله اعلی</u>
شماره دانشجویی: <u>960164039</u>	سرپرست کارآموزی: <u>مهندس امیرحسین امانی</u>
رشته / گرایش: <u>مهندسی کامپیوتر / نرم افزار</u>	ترم و کد کارآموزی: <u>ترم پاییز سال 1399 - 1400</u>
محل کارآموزی: <u>شرکت فناوری اطلاعات دانش دیجیتال</u>	موضوع کارآموزی: <u>1884156372</u>
<p>فعالیت های انجام شده: <u>یادگیری (سیدیلین) TDD</u> <u>بررسی مجدد</u> <u>AVC - Abstraction</u> <u>تجزیه و تحلیل</u> <u>گراف</u> <u>و سایر موارد</u></p> <p>فعالیت های آتی: <u>در ادامه به یادگیری (سیدیلین) این (سیدیلین) بررسی (سیدیلین) هم در زمینه یادگیری گراف و سایر موارد مشکلات:</u></p>	
<p>پیشنهادهای: <u>قبل از یادگیری (این (سیدیلین) تمامی کدهای بین استاتی بودند و وقتی به یادگیری (این (سیدیلین) رسیدیم تغییراتی در کدها</u></p> <p><u>نمودار (این (سیدیلین) اما بعد از یادگیری (این (سیدیلین) و یادگیری (این (سیدیلین) در مرحله API ها</u></p> <p><u>مستندات (این (سیدیلین) و در ادامه حل آن و کما کما هم برای (این (سیدیلین) و هم برای (این (سیدیلین) و هم برای (این (سیدیلین)</u></p> <p><u>(این (سیدیلین) و در ادامه حل آن و کما کما هم برای (این (سیدیلین) و هم برای (این (سیدیلین) و هم برای (این (سیدیلین)</u></p> <p>نظریه سرپرست کارآموزی: <u>تاریخ و امضاء دانشجو:</u></p>	
<p>نظریه استاد کارآموزی: <u>تاریخ و امضاء استاد کارآموزی:</u></p>	

فرم شماره (3)

دانشگاه آزاد اسلامی
واحد تهران مرکزی

فرم پایان دوره کارآموزی

نام و نام خانوادگی دانشجو: <u>یلین کیم</u>	استاد کارآموزی: <u>(سرایان) عطاردزاد</u>
شماره دانشجویی: <u>960164039</u>	سرپرست کارآموزی: <u>امیرحسین امای</u>
رشته / گرایش: <u>مهندسی کامپیوتر / نرم افزار</u>	ترم و کد کارآموزی: <u>ترم پاییز سال 1399-1400</u>
محل کارآموزی: <u>سازمان نوآوری و نوآوری های دانشی</u>	موضوع کارآموزی: <u>یادگیری و انجام پروژه های آموزشی در سیستم های</u>

نظریات سرپرست کارآموزی	عالی 4نمره	خوب 3نمره	متوسط 2نمره	ضعیف 1نمره
حضور و غیاب و رعایت نظم و ترتیب در واحد صنعتی	✓			
میزان علاقه به همکاری و فراگیری	✓			
کمب تجربه کاری و بکارگیری تکنیک ها	✓			
ارزش پیشنهادات کارآموز جهت بهبود کار	✓			
کیفیت گزارش های کارآموزی به واحد صنعتی	✓			

<p>توضیح:</p> <p>امضاء سرپرست کارآموزی:</p> <p>تاریخ: <u>۹۹/۱/۵</u></p> <p>نمره نهایی به حروف: <u>بسیار</u></p> <p>نمره نهایی به عدد: <u>۴۵</u></p>	<p>پیشنهادات سرپرست کارآموزی جهت بهبود برنامه کارآموزی</p> <p><u>مسئله های در این زمینه استفاده از تکنیک</u></p> <p><u>در این زمینه</u></p> <p><u>در این زمینه</u></p> <p>امضاء سرپرست کارآموز:</p> <p>تاریخ: <u>۹۹/۱/۵</u></p>
---	---



شرکت دانش بنیان نوآوران ورزش دیجیتال

بسمه تعالی

شماره: ۱۳۹۹-۱۱-۵۰۱

تاریخ: ۱۳۹۹/۱۱/۰۵

پیوست: ندارد

از: شرکت نوآوران ورزش دیجیتال
به: دانشگاه آزاد اسلامی واحد تهران مرکزی

با سلام و احترام:

احتراما بدین وسیله گواهی می شود که بازگشت به نامه شماره ۸۰۹/۹۰۲۹۵ مورخ ۱۳۹۹/۱۱/۵ سرکار خانم نیکی نجفی به شماره دانشجویی ۹۶۰۱۶۴۰۳۹ دانشجوی رشته مهندسی کامپیوتر-ترم افزار از تاریخ ۱۳۹۹/۱۱/۱ لغایت ۱۳۹۹/۱۱/۲۰ به مدت ۲۰ روز جمعا ۱۲۰ ساعت کار مفید، دوره کارآموزی خود را به سرپرستی جناب آقای مهندس امیرحسین امانی در این شرکت آغاز نموده است. این گواهی صرفا جهت ارائه به دانشگاه مذکور صادر شده و فاقد هرگونه ارزش قانونی دیگر است.

با احترام فراوان
رئیس هیئت مدیره
امیرحسین امانی
نوآوران ورزش دیجیتال
۹۹/۱۱/۵



شرکت دانش بنیان نواوران ورزش دیجیتال

بسمه تعالی

شماره: ۱۳۹۹-۱۱-۵۰۱

تاریخ: ۱۳۹۹/۱۱/۰۵

پیوست: ندارد

از: شرکت نواوران ورزش دیجیتال
به: دانشگاه آزاد اسلامی واحد تهران مرکزی

با سلام و احترام:

احتراما بدین وسیله گواهی می شود که بازگشت به نامه شماره ۸۰۹/۹۰۲۹۵ مورخ ۱۳۹۹/۱۱/۵ سرکار خانم نیکی نجفی به شماره دانشجویی ۹۶۰۱۶۴۰۳۹ دانشجوی رشته مهندسی کامپیوتر-ترم افزار از تاریخ ۱۳۹۹/۱۱/۱ لغایت ۱۳۹۹/۱۱/۲۰ به مدت ۲۰ روز جمعا ۱۲۰ ساعت کار مفید، دوره کارآموزی خود را به سرپرستی جناب آقای مهندس امیرحسین امانی در این شرکت به پایان برده است. این گواهی صرفا جهت ارائه به دانشگاه مذکور صادر شده و فاقد هرگونه ارزش قانونی دیگر است.

با احترام فراوان
رئیس هیئت مدیره
امیرحسین امانی
وزش دیجیتال
۹۹/۱۱/۵

تاریخ ۱۳۹۹/۱۱/۰۱

شماره ۸۰۹/۹۰۲۹۵

پیوست

دانشگاه آزاد اسلامی



واحد تهران مرکزی

باسمه تعالی

معرفی نامه کارآموزی

اول

۱۳۹۹-۱۴۰۰

به : شرکت نوآوران ورزش دیجیتال

از : دانشگاه آزاد اسلامی واحد تهران مرکزی

موضوع : معرفی کارآموز

سلام علیکم

بدینوسیله خانم / آقای نیکی نجفی دانشجوی رشته مهندسی کامپیوتر مقطع کارشناسی به شماره دانشجویی ۹۶۰۱۶۴۰۳۹ را جهت گذراندن دوره کارآموزی از تاریخ ۱۳۹۹/۱۰/۳۰ به مدت ۲ ماه جمعاً ۲۰ ساعت کارمقد معادل ۱ واحد درسی اجباری معرفی می نماید.

استاد راهنمای کارآموزی ایشان خانم / آقای دکتر ایمان عطارزاده می باشد.

خواهشمند است دستور فرمایید ضمن اقدامات لازم به منظور اعلام کتبی شروع کارآموزی نامبرده نسبت به تعیین سرپرست کارآموزی اقدام مقتضی صورت گیرد. متمنی است پس از انجام دوره کارآموزی، گواهی پایان دوره را جهت نامبرده صادر و یک نسخه از آنرا به دفتر این دانشکده ارسال فرمایند.

ضمناً لازم به ذکر است دانشجوی در طول مدت دوره کارآموزی تحت پوشش بیمه حوادث سازمان مرکزی دانشگاه می باشد و نامبرده موظف به اجرای مقررات انضباطی و ایمنی آن واحد صنعتی می باشد.

رونوشت :

۱ - استاد راهنما جهت استحضار

۲ - دانشجو



نشانی : تهران - خ اکندری شمالی - خ فرصت شیرازی - پلاک ۱۳۶ صندوق پستی ۱۳۱۸۵/۷۶۸ تلفن: ۶۶۹۳۳۵۰-۳ و ۸۴ فکس: ۶۶۴۳۶۶۸۰ نمابر: ۶۶۲۸۹۷۲

www.iauctb.ac.ir

دانشگاه آزاد اسلامی

واحد تهران مرکزی



شماره:

تاریخ:

پیوست:

باسمه تعالی

ریاست محترم دفتر ارتباط با صنعت:

رشته مهندسی کامپیوتر

بدینوسیله گزارش کارآموزی خانم/آقای دانشجو سینا کجفی

مقطع کارشناسی به شماره دانشجویی: 960164039

گرایش نرم افزار

با موضوع کارآموزی: یا (سینا) وائی (گروه پرسپلیس) (رسیدین) TDD

که تحت راهنمایی استاد: ایمان معصا زرار به اتمام رسیده تایید می گردد. مدارک به همراه دو عدد CD

هر کدام حاوی فایل های WORD و PDF گزارش کامل کارآموزی و SCAN مدارک (جهت ارسال به دفتر ارتباط با صنعت)

مورد تایید می باشد.

مدیر گروه آموزشی

استاد راهنما

امضاء

امضاء

آدرس: تهران - خ آزادی - خ اسکندرس شمالی - خ فرحت شیرازی - پلاک ۱۶۰ - سب پ ۷۶۸/۱۳۱۸۵
تلفن: ۶۶۱۳۶۶۸۲ - ۶۶۱۳۶۶۸۲ - ۶۶۱۳۵۵۸۲ - فاکس: ۶۶۱۳۵۵۸۲ - آدرس اینترنتی: www.IAUCTB.ORG

مدارک لازم:

- ☐ اصل فرم های ۱، ۲ و ۳ کارآموزی
- ☐ کپی معرفی نامه دانشکده به شرکت/کارخانه دوره کارآموزی
- ☐ اصل نامه شروع به کار - کارآموزی با مهر رسمی شرکت/کارخانه دوره کارآموزی (عطف به شماره معرفی نامه دانشجو)
- ☐ اصل نامه پایان کار - کارآموزی با مهر رسمی شرکت/کارخانه دوره کارآموزی (عطف به شماره معرفی نامه دانشجو)
- ☐ ۲ نسخه CD هر کدام حاوی فایل های WORD و PDF گزارش کامل کارآموزی و SCAN مدارک فوق جهت دفتر ارتباط با صنعت
- ☐ نسخه سوم از CD فوق جهت استاد راهنمای کارآموزی

در صورت کامل بودن مدارک، پس از تایید گزارش کارآموزی (حداقل ۶۰ صفحه متن گزارش) توسط استاد راهنما، ثبت نمره دانشجو بلامانع می باشد.