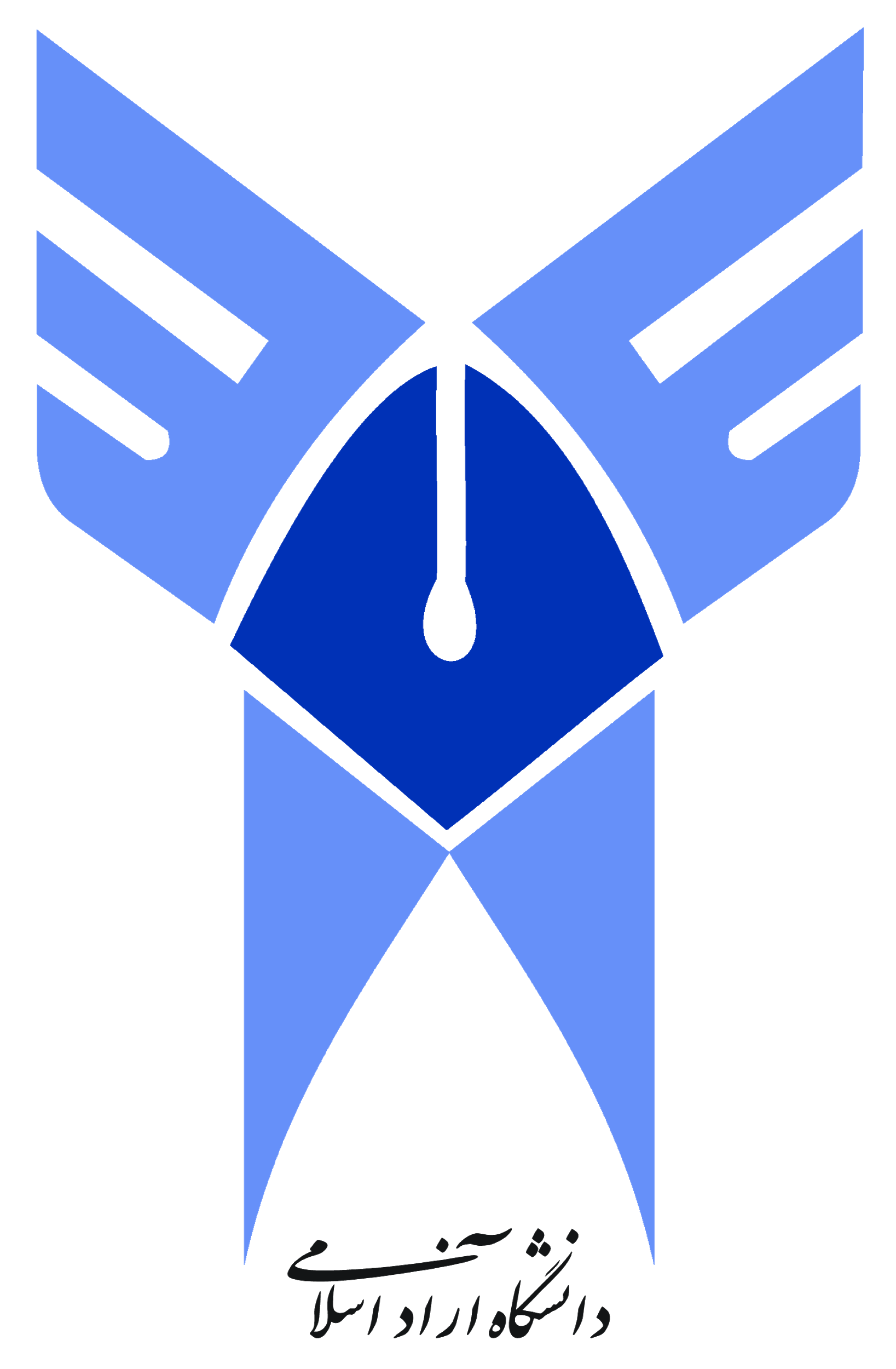
****

**دانشگاه آزاد اسلامی واحد تهران مرکزی**

**دانشکده‌ی فنی و مهندسی**

**مجتمع آیت الله هاشمی رفسنجانی (ره)**

**گروه آموزشی مهندسی کامپیوتر**

**گزارش پروژه‌ی کارشناسی**

**عنوان:**

**پیاده‌سازی سامانه رزرو و ثبت زمان‌بندی‌های دفاع پایان‌نامه‌ها**

**استاد راهنما:**

**جناب آقای دکتر محسن جهانشاهی**

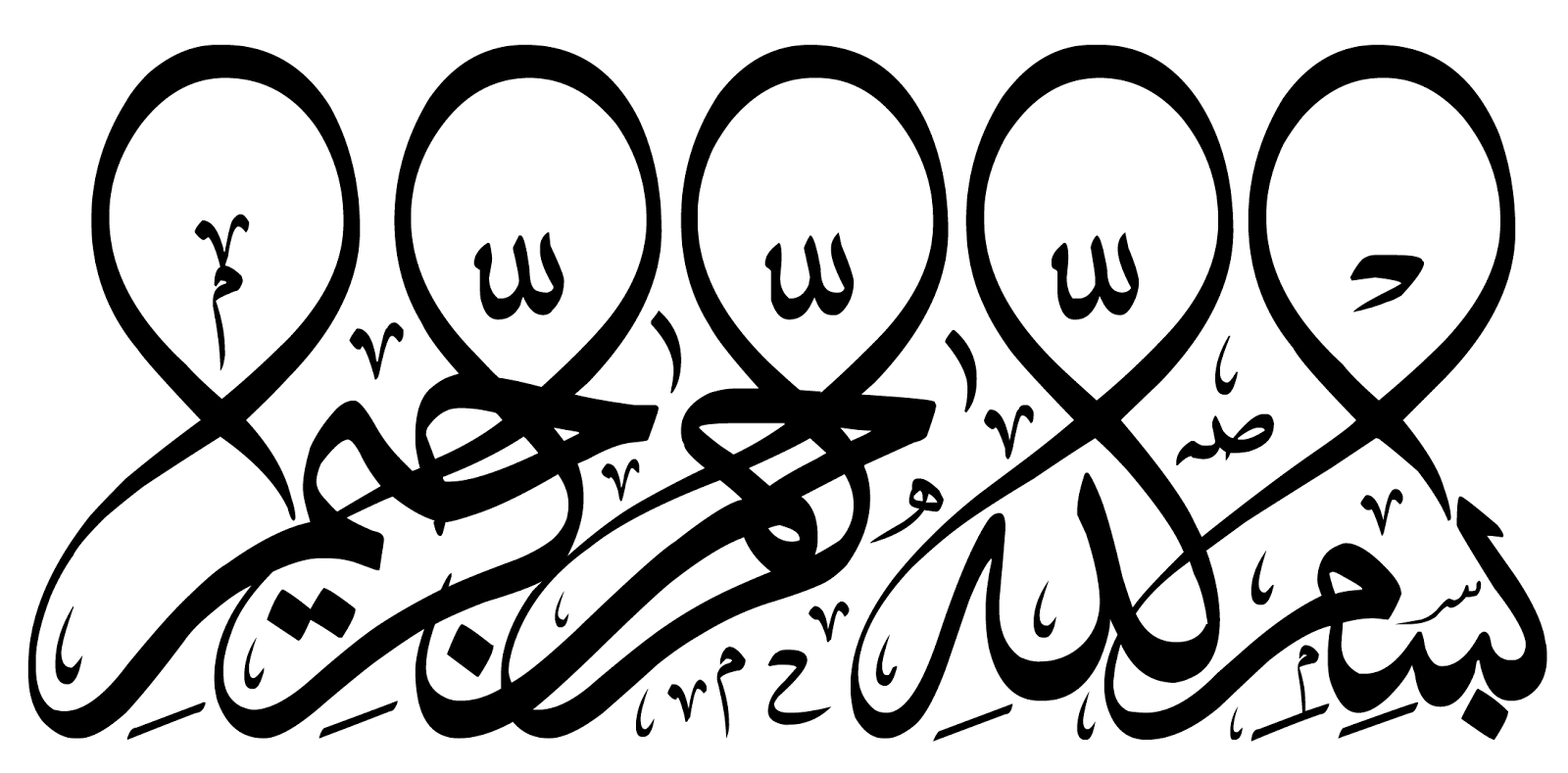
**نگارش:**

**سید اشکان طراوتی**

**دانشجوی کارشناسی مهندسی کامپیوتر گرایش نرم‌افزار**

**شماره دانشجویی: ۹۳۰۲۲۵۸۲۰**

**نیمسال دوم ۹۷-۱۳۹۶**

****

**قدردانی**

از استاد گرانقدر جناب آقای دکتر جهانشاهی هم به عنوان استاد راهنما و هم به عنوان مدیریت گروه مهندسی کامپیوتر بسیار سپاسگزارم که همواره در چالش‌های گوناگون رهنما و آموزگار بنده بوده اند و بدون راهنمایی‌های ایشان پیشبرد این پروژه بسیار مشکل می‌نمود.

همچنین از اساتید گرامی جناب آقای دکتر وزیری بابت راهنمایی‌ها و تعلیم بی‌دریغ ایشان در طراحی پایگاه داده‌ها، جناب آقای دکتر نصیری بابت راهنمایی‌های ایشان در طراحی و معماری نرم‌افزار، جناب آقای مهندس علامه بابت تدریس رهیافت جدید شبکه، جناب آقای مهندس تناوش بابت تلاش‌های ایشان در آشنایی ما با همروندی و مشکلات مربوط به آن، جناب آقای مهندس عابدسلیمی بابت تدریس بی‌نقص ایشان در رابطه با سیستم عامل لینوکس که عصای دست بنده در این پروژه بود، جناب آقای دکتر هاشمی طبا بابت تذکرات و نکات مهمی که ایشان در مبحث مهندسی نرم‌افزار به بنده و هم‌شاگردی‌ها دادند و در نهایت سرکار خانم دکتر اصغری که در کنار تعلیم همیشه راهنما و مشوق همگی ما بوده اند .

همچنین جا دارد از دوستانی چون آقای مهندس شالباف‌زاده، آقای مهندس ایرانشاهی، خانم مهندس دشتی و خانم مهندس قمی که در روند این پروژه بارها به بنده کمک کردند نیز تشکر و قدردانی نمایم.

از پدر و مادر عزیزم سپاسگزارم که در این مدت مشغله‌ی بنده را تحمل کردند.

**فهرست مطالب**

[1 مقدمه و معرفی سیستم 3](#_Toc522812729)

[1.1 شناخت سیستم و صورت مسئله 3](#_Toc522812730)

[1.2 ویژگی‌های سیستم 3](#_Toc522812731)

[2 مروری بر مفاهیم و شاکله‌ی علمی سیستم 4](#_Toc522812732)

[2.1 چرخه‌ی حیات توسعه‌ی نرم‌افزار 4](#_Toc522812733)

[2.2 نرم‌افزار کاربردی 10](#_Toc522812734)

[2.3 سامانه‌ی اطلاعاتی 11](#_Toc522812735)

[2.4 لایه‌ی کاربرد شبکه و وب 11](#_Toc522812736)

[2.5 معماری client/server 13](#_Toc522812737)

[2.6 نرم‌افزار تحت وب 18](#_Toc522812738)

[2.7 معماری نرم‌افزار و الگوی معماری 19](#_Toc522812739)

[2.8 معماری MVC و تفکیک دغدغه‌ها 21](#_Toc522812740)

[2.9 فریم‌ورک 21](#_Toc522812741)

[2.10 مدلسازی نرم‌افزار 21](#_Toc522812742)

[2.11 مستند‌های متداول پروژه‌های نرم‌افزاری 22](#_Toc522812743)

[2.12 اصل DRY در توسعه‌ی نرم‌افزار 22](#_Toc522812744)

[2.13 حمله‌ی CSRF 23](#_Toc522812745)

[2.14 برنامه‌نویسی شیءگرا 24](#_Toc522812746)

[2.15 نگاشت داده‌ها (Data Mapper و ORM) 25](#_Toc522812747)

[3 معرفی تکنولوژی‌های به کار رفته در توسعه‌ی سیستم 27](#_Toc522812748)

[3.1 ابزار کنترل نسخه‌ی کد منبع گیت 27](#_Toc522812749)

[3.2 سیستم عامل گنو/لینوکس 27](#_Toc522812750)

[3.3 زبان پایتون 28](#_Toc522812751)

[3.4 زبان جاوااسکریپت 32](#_Toc522812752)

[3.5 فریم‌ورک جنگو 36](#_Toc522812753)

[3.6 فریم‌ورک بوت‌استرپ 37](#_Toc522812754)

[3.7 تکنولوژی AJAX 38](#_Toc522812755)

[3.8 پایگاه داده Sqlite 40](#_Toc522812756)

[4 مستندات فنی پروژه 40](#_Toc522812757)

[4.1 مدلسازی سیستم 40](#_Toc522812758)

[4.2 تشریح اجزای سیستم 44](#_Toc522812759)

[4.3 ساختار پروژه‌های جنگو 44](#_Toc522812760)

[4.4 نمونه کد‌های سمت سرور 51](#_Toc522812761)

[4.5 نمونه کد‌های سمت کلاینت 55](#_Toc522812762)

[4.6 روش راه‌اندازی سامانه در محیط توسعه 62](#_Toc522812763)

[5 راهنمای کاربری سامانه 64](#_Toc522812764)

[5.1 راهنمای بخش جدول زمان‌بندی 64](#_Toc522812765)

[5.2 راهنمای بخش دانشجو 66](#_Toc522812766)

[5.3 راهنمای بخش مدیریت 69](#_Toc522812767)

[6 منابع و مراجع 72](#_Toc522812768)

**فهرست اشکال**

[تصویر 1 توصیف بصری چرخه‌ی حیات توسعه‌ی نرم‌افزار 4](#_Toc522812800)

[تصویر 2نمود تصویری عملکرد AJAX 39](#_Toc522812801)

[تصویر 3 نمودار موارد کاربرد (Use Case) 41](#_Toc522812802)

[تصویر 4 نمودار کلاس 42](#_Toc522812803)

[تصویر 5 دیاگرام موجودیت‌بستگی 43](#_Toc522812804)

[تصویر 6 نمودار موجودیت بستگی جداول فریم‌ورک 44](#_Toc522812805)

[تصویر 7 ساختار دایرکتوری محیط توسعه یا استقرار 45](#_Toc522812806)

[تصویر 8 ساختار دایرکتوری اصلی پروژه 46](#_Toc522812807)

[تصویر 9 ساختار پوشه‌ی مربوط به پیکربندی 47](#_Toc522812808)

[تصویر 10ساختار یک app 51](#_Toc522812809)

[تصویر 11 صفحه‌ی اصلی سامانه 64](#_Toc522812810)

[تصویر 12 جدول زمانبندی جلسات 65](#_Toc522812811)

[تصویر 13 اعلامیه‌ی دفاع 65](#_Toc522812812)

[تصویر 14 چاپ اعلامیه 66](#_Toc522812813)

[تصویر 15 صفحه‌ی ورود 66](#_Toc522812814)

[تصویر 16 داشبورد دانشجو 67](#_Toc522812815)

[تصویر 17 داشبورد دانشجو - پیگیری درخواست 67](#_Toc522812816)

[تصویر 18صفحه‌ی زمان‌ها قابل رزرو 68](#_Toc522812817)

[تصویر 19 لغو رزرو 68](#_Toc522812818)

[تصویر 20 داشبورد مدیریت 69](#_Toc522812819)

[تصویر 21 رویداد‌های اخیر 69](#_Toc522812820)

[تصویر 22 مدیریت کاربران 70](#_Toc522812821)

[تصویر 23 ویرایش کاربر 70](#_Toc522812822)

[تصویر 24 گروه‌ها (نقش‌ها) 71](#_Toc522812823)

[تصویر 25 مدیریت دانشجویان 71](#_Toc522812824)

مقدمه و معرفی سیستم

شناخت سیستم و صورت مسئله

این نرم‌افزار بر بستر وب به دانشجویان تحصیلات تکمیلی گروه‌های آموزشی و دفتر پژوهش دانشکده اجازه می‌دهد تا در تعامل با یکدیگر به آسانی اتاق دفاعیه را برای جلسه‌ی دفاع و حضور اساتید راهنما،مشاور و داور رزرو و هماهنگ نمایند.

کاربران اصلی و عمده‌ی این سامانه دانشجویان و پرسنل پژوهش می‌باشند.

ویژگی‌های سیستم

مطابق با موضوع پروژه و شناخت ارائه‌شده، این مستند به تشریح مراحل استحصال «**سامانه رزرو و ثبت زمان‌بندی‌های دفاع پایان‌نامه‌ها**» می‌پردازد. این محصول یک نرم‌افزار کاربردی از نوع سامانه‌ی اطلاعاتی تحت وب می‌باشد که با استفاده از تکنولوژی‌های مدرن توسعه‌ی وب به کمک زبان‌های برنامه‌نویسی پایتون و جاوااسکریپت بر پایه‌ی فریم‌ورک‌های جنگو (یک فریم‌ورک سمت سرور مبتنی بر الگوی معماری MVC) و بوت‌استرپ پیاده‌سازی شده و هدف آن تسهیل امور برنامه‌ریزی و تخصیص اتاق‌های دفاع برای برگزاری جلسات دفاع دانشجویان مقاطع تحصیلات تکمیلی می‌باشد. این مستند ترکیبی از چند نوع مستند متداول نرم‌افزار است و شامل مدلسازی و قطعه‌کد می‌باشد.

با توجه به موارد فوق ویژگی‌های زیر برای این سامانه مطرح خواهد بود:

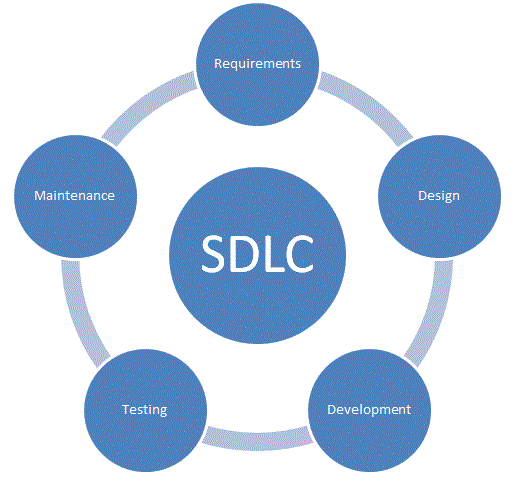
* نگه‌داری‌پذیری
* انعطاف و قابلیت اضافه کردن امکانات بیشتر
* استقرار آسان
* توزیع‌پذیری
* مقیاس‌پذیری نه چندان دشوار
* ساختار ساده و قابل مطالعه
* استفاده از تکنولوژی‌های مدرن و پایدار
* دسترسی آسان بر بستر وب
* تجربه‌ی کاربری مناسب
* سازگار با توسعه‌ی چابک
* استفاده از امکانات مدرن مرورگر‌های وب و واکنش‌گرایی

مروری بر مفاهیم و شاکله‌ی علمی سیستم

در این قسمت به طور اجمالی به معرفی و بررسی زیرساخت‌های علمی سامانه می‌پردازیم. در حقیقت این قسمت بیان می‌دارد که از کدام آموزه‌های مهندسی کامپیوتر در اجرای این پروژه استفاده شده است.

چرخه‌ی حیات توسعه‌ی نرم‌افزار

این مبحث در مهندسی نرم افزار و رشته‌های مرتبط با موضوع سیستم‌های اطلاعاتی مورد استفاده قرار می‌گیرد و موضوع آن تشریح فرآیندهای مرتبط با برنامه‌ریزی و هدف گذاری، تحلیل، تولید، آزمایش، استقرار و نگهداری سیستم‌های نرم افزاری است که اغلب در یک چرخه مانند آنچه در دیاگرام مشاهده می‌شود تداوم و تکرار پیدا می‌کنند. این چرخه همچنین می‌تواند در برگیرنده فرایندهای مربوط به تأمین سخت افزار یا ترکیبی از سخت افزارها و نرم افزارها نیز دانسته شود.



تصویر 1 توصیف بصری چرخه‌ی حیات توسعه‌ی نرم‌افزار

كليه پروژه هاي نرم افزاري الزامي است داراي يك رهيافت چرخه حيات باشند:

* مرحله UR - بيان نيازهاي كاربر User Requirements
* مرحله SR - بيان نيازهاي نرم افزار Software Requirements
* مرحله AD - طراحي معماري Architectural Design
* مرحله DD - طراحي تفصيلي و توليد برنامه Detailed Design
* مرحله TR - انتقال و واگذاري نرم افزار براي بهره برداري Transfer of the software
* مرحله OM - بهره برداري و نگهداري Operations & Maintenance

چهار مرحله اول با يك بازبيني كه بوسيله نشانه “R/” نمايش داده شده است خاتمه مي يابند (به عنوان مثال UR/R بازبيني نيازهاي كاربر است). خواه پروژه توسط كاركنان داخلي و يا از طريق شركتها و پيمانكاران صنعت

نرم افزار انجام شود، اين مراحل با توجه به اندازه، كاربرد (مثلا علمي،‌اداري، بلادرنگ، دسته اي)، سخت افزار، سيستم عامل يا زبان برنامه نويسي استفاده شده انجام مي پذيرند، هر چند كه هر يك از اين عوامل رهيافت توليد، شيوه ومحتوي اقلام تحويل دادني را تحت تاثير قرار مي دهند.

تعيين نيازهاي كاربر - مرحله UR

مرحله UR مرحله تعريف مسئله در يك پروژه نرم افزاري است.

* دامنه و وسعت سيستم بايد مشخص گردد.
* نيازهاي كاربر بايد تعيين گردد. اين امر مي تواند بوسيله مصاحبه، بازديد و بررسي و يا ساخت نمونه هاي اوليه صورت پذيرد.
* خواسته هاي مشخص كاربر بايد تعيين شده و در سند نيازهاي كاربر (URD) نوشته شوند.

پيش از تكميل بازبيني نيازهاي كاربر (UR/R) طرح مديريت پروژه نرم افزار حاوي رئوس مطالب كل پروژه بايد بوسيله توليد كننده نرم افزار تهيه شود. أين طرح بايد شامل برآورد هزينه پروژه باشد. همچنين طرحهاي تفصيلي تر براي مرحله تعيين نيازهاي نرم افزار نيز بايد تهيه شوند.

تعيين نيازهاي نرم افزار - مرحله SR

* مرحله SR مرحله تحليل پروژه نرم افزاري است.
* يك بخش ضروري فعاليت تحليل، ساخت ” الگويي” است كه بيانگر آن باشد كه نرم افزار چه كاري را بايد انجام دهد،‌نه اينكه چگونه بايد آن را انجام دهد. در اين حالت ممكن است ساخت نمونه هاي اوليه به منظور روشن نمودن نيازهاي نرم افزار ضروري باشد.

تحويل دادني اصلي در اين مرحله سند نيازهاي نرم افزار (SRD) است . SRD هميشه بايد برا ي هر پروژه نرم افزاري توليد گردد. اصطلاحات پياده سازي مي بايست از SRD حذف شود. أين سند بايد رسما” توسط كاربر، مهندسين نرم افزار و سخت افزار كامپيوتر و مديران ذيربط در حين بازبيني نيازهاي نرم افزار (SR/R) مورد بررسي قرار گيرد.

در طول مرحله SR در قسمت طرح مديريت پروژه نرم افزار، رئوس مطالب باقيمانده پروژه بايد بروز آورده شود.

طرح بايد شامل برآورد مجموع هزينه هاي پروژه باشد. همچنين طرحهاي تفصيلي تر براي مرحله طراحي معماري نيز بايد تهيه شود.

طراحي معماري - مرحله AD

هدف مرحله طراحي معماري (AD) تعيين ساختار نرم افزار است.

* الگوي ساخته شده در مرحله SR نقطه شروع اين مرحله است. اين الگو با تخصيص كاركردها به مولفه هاي نرم افزار و تعيين گردش اطلاعات و عمليات بين آنها به طرح معماري نرم افزار تغيير مي يابد.
* در اين مرحله ممكن است طرح چندين بار تكرار شود. مشكلات تكنيكي و يا قسمتهاي بحراني و حساس طرح بايد مشخص گردد. ممكن است نمونه سازي بخشهاي حساس نرم افزار جهت تایيد فرضيات طرح اصلي ضروري باشد. در اين مرحله طرح‌هاي جانشين نيز مي توان در نظر گرفت كه نهايتا”‌يكي از آنها بايد انتخاب شود.

عنصر تحويل دادني كه خروجي اصلي اين مرحله به شمار مي آيد، سند طراحي معماري (ADD) است. ADD همواره بايد براي هر پروژه نرم افزاري توليد گردد. ADD بايد رسما” بوسيله مهندسين نرم افزار و سخت افزار كامپروتر، كاربران و مديران ذيربط در حين بازبيني طرح معماري (AD/R) مورد بررسي قرار گيرد.

* در طول مرحله AD طرح مديريت پروژه نرم افزار حاوي رئوس مطالب باقيمانده بايد تهيه شود. اين طرح بايد شامل برآورد هزينه پروژه (حداكثر 10% خطا بسيار مطلوب است) باشد. همچنين طرحهاي جزئي تر براي مرحله طراحي تفصيلي ( DD ) نيز مي بايد تهيه شود.

طراحي تفصيلي و توليد - مرحله DD

هدف اين مرحله طراحي تفصيلي نرم افزار، برنامه نويسي، مستندسازی و آزمون آن است.

سند طراحي تفصيلي ( DDD: Detailed Design Document ) و راهنماي كاربر نرم افزار ( SUM: Software User Manual ) همزمان با برنامه نويسي و آزمون آن توليد مي شود. در آغاز DDD و SUM بخشهاي متناظر با سطوح فوقاني سيستم را در بر مي گيرند. در پايان اين مرحله، مستندات تكميل مي شوند و به همراه برنامه ها عناصر تحويل دادني اين مرحله را تشكيل مي دهند.

• در طول اين مرحله فعاليت هاي آزمون واحد،‌آزمون يكپارچگي و آزمون سيستم بر طبق طرح‌هاي مصوب تایيد شده در مراحل SR و AD اجرا مي گردند. به موازات اين آزمون ها مي بايست كيفيت نرم افزار نيز مورد سنجش قرار گيرد.

سه عنصر تحويل دادني (برنامه، DDD و SUM) كه مستلزم بازبيني هاي مرحله‌اي در طول مرحله DD مي باشند بايد رسما” توسط مهندسين نرم افزار و مديریت ذيربط در حين بازبيني طرح تفصيلي (DD/R) مورد بررسي قرار گيرد. در پايان فرآيند بازبيني، نرم افزار جهت آزمونهاي پذيرش موقت آماده است.

انتقال - مرحله TR

هدف اين مرحله تایيد آن است كه نرم افزار كليه خواسته هاي كاربر را كه در سند نيازهاي كاربر (URD) مطرح و تنظيم گرديده اند، برآورد مي نمايد. اين امر بوسيله نصب نرم افزار و اجراي آزمون پذيرش انجام

مي شود.

زماني كه نرم افزار جهت ارائه قابليتهاي مورد نياز نمايش داده مي شود،‌نرم افزار مي تواند موقتا” مورد پذيرش قرار گرفته و بهره برداري از آن آغاز شود.

سند انتقال نرم افزار (STD) بايد به منظور مستند ساختن انتقال نرم افزار به گروه بهره برداري در حين مرحله TR تنظيم شود.

بهره برداري و نگهداري - مرحله OM

زماني كه نرم افزار آماده بهره برداري مي گردد، مي بايست به منظور حصول اطمينان از آنكه كليه خواسته هاي تعيين شده در سند نيازهاي كاربر را برآورده مي سازد به دقت مورد بررسي قرار گيرد. برخي از خواسته ها به مدت زماني جهت مشخص شدن نياز دارند.

هنگامي كه نرم افزار كليه آزمونهاي پذيرش را با موفقيت گذراند، مي تواند مورد پذيرش نهايي قرار گيرد.

سند تاريخچه پروژه ( PHD: Project History Document ،) اطلاعات مديريتي مهم جمع آوري شده در طول جريان پروژه را خلاصه مي نمايد.اين سند بايد پس از پذيرش نهايي صادر شود. اين سند مي بايست در پايان چرخه حيات، به همراه ساير اطلاعات گردآوري شده در مرحله بهره برداري و نگهداري مجددا” منتشر و اعلام گردد.

* پس از پذيرش نهايي،‌نرم افزار بايد به منظور تصحيح خطاهاي آشكار شده در طول مراحل قبل و يا به علت نيازهاي جديدي كه رخ مي‌دهند، اصلاح گردد. اين عمل ” نگهداري ” ناميده مي شود.
* در تمام دوران بهره برداري مي بايست توجه ويژه أي جهت به هنگام نگهداشتن مستندات اعمال گردد. اطلاعات خطاها و نواقص مي بايست به منظور تهيه داده هاي خام براي تعيين معيارهاي كيفيت نرم افزار در پروژه هاي آتي، ضبط و ثبت گردد. همچنين مي بايست از ابزار لازم به منظور تسهيل در جمع آوري و تحليل داده هاي كيفيت نرم افزار استفاده شود.

چرخه حیات توسعه نرم افزار به سه فاز اصلی و یک فاز چتری(پشتیان) تقسیم بندی می شود. که فازهای اصلی عبارتند از:

1. تعریف
2. توسعه
3. نگهداری

هر یک از فازها خود به فازهایی تقسیم می شوند.

1- فاز تعریف

در مرحله تعريف تاكيد بر چه (WHAT) است:

چه توابع و عملكردي انتظار مي رود؟

چه رفتاري بايد سيستم داشته باشد؟

چه واسط هاي كاربري بايد طراحي شود؟

چه محدوديت هايي براي طراحي وجود دارد؟

معيارهاي صحت اعتبار براي تعريف يك نرم افزار موفق چيست؟

در اين مرحله روش مورد استفاده با انجام سه فعاليت زير مشخص مي شود:

مهندسي سيستم يا اطلاعات System/Information Engineering

برنامه ريزي پروژه نرم افزاري Software Project Planning

تحليل نيازها Requirements' Analysis

2- فاز توسعه

مرحله توسعه تاكيد بر چگونگي (HOW) دارد:

چگونه داده ها بايد سازمان دهي شوند؟

تكميل وظايف چگونه در معماري وظايف گنجانده مي شود؟

ارتباط و توصيف پردازشها چگونه انجام مي گيرد؟

ويژگي هاي واسط هاي كاربري چيست؟

چگونه آزمايش انجام مي گيرد؟

چگونه طراحي تبديل به معماري برنامه و برنامه نويسي مي گردد؟ (مهمترين فعاليت)

در اين مرحله مدل هاي مورد نياز توليد مي گردد و روند كلي برنامه ها تعيين شده و ساختار پايگاه داده ها نيز رسم مي شود.

سه فعاليت در اين مرحله عبارتند از:

طراحي نرم افزار Software Design

توليد كد Coding

تست Testing

3- فاز نگهداري

مرحله سوم مرحله نگه داري است كه شامل:

نگه داري اصلاحي (Correction): اشكالات در چرخه زندگي پروژه شناسايي، بررسي و رفع مي‌گرددند.

نگهدار تطبيقي (Adaptation): تغيير به مرور زمان و با توجه به نيازهاي محيطي و كاربر.نياز جديد user را بايد با محيط تطبيق داد منوط به اينكه نرم افزار قابليت توسعه را داشته باشد.

نگه داري ارتقايي (Enhancement): ارتقا نرم افزار پس از راه اندازي كه توسط كاربر خواسته مي شود.

نگه داري پيشگيرانه (Prevention): ايجاد تغييراتي در برنامه ها كه اصلاح و بهبود نرم افزار را تسهيل مي كند.

نرم افزار پس از دوره مشخصي بازنگري شده و اصلاحات لازم انجام مي شود، درواقع نرم افزار جديدي توليد مي گردد.

فعاليت هاي چتری (پشتيباني) :

فعاليت هاي پشتيباني زير در كليه مراحل فوق انجام مي گردد:

كنترل و نظارت بر برنامه ريزي پروژه

بازنگري هاي مستمر فني و رسمي (Quality Control)

اطمينان مرغوبيت نرم افزار ( شامل تمامي مراحل ) Quality Assurance

مديريت پيكربندي نرم افزار Configuration/Change Management

تهيه و تدوين مستندات

اندازه گيري ( مقياسهاي اندازه گيري ) Software Measurements

مديريت ريسك ( پيامدهايي كه برآورد زمان و هزينه را تغيير دهد) Risk Management

نرم‌افزار کاربردی

نرم‌افزار کاربردی که در زبان انگلیسی به آن Application software می‌گویند، عبارت است از نرم‌افزاری که با استفاده از منابع و قابلیت‌های رایانه کاری را مستقیما برای کاربر انجام می‌دهد. باید توجه داشت که این عبارت در مقابل عبارت نرم‌افزار سیستمی معنی پیدا می‌کند.

نرم‌افزار سیستمی در مقابل در پس‌زمینه عمل می‌کند و خدماتی را فراهم می‌کند که دیگر نرم‌افزارها یا سیستم‌عامل می‌توانند برای انجام کارهای خود از آن استفاده کنند. اما در عوض معمولا مستقیما با کاربر عادی در تماس نیست و خدماتی را به او ارائه نمی‌دهد.

در عمل، بیشتر نرم‌افزارهایی که کاربران با آنها سر و کار دارند از این دسته محسوب می‌شوند. برای مثال می‌توان به نرم‌افزارهای رومیزی یعنی واژه‌پردازها، آفیس، نرم‌افزارهای طراحی گرافیکی، بازی‌های رایانه‌ای و امثال آنها اشاره کرد.

بسیاری از نرم‌افزارهای کاربردی، برای توسعه‌دهندگان ابزار رابط برنامه‌نویسی کاربردی را فراهم می‌کنند تا بتوان از قابلیت‌های نرم‌افزار در نرم‌افزارهای جدید استفاده کرد. برای مثال نرم‌افزار ادوبی آکروبات هنگام نصب، ای‌پی‌آی فراهم می‌کند که برنامه‌نویس می‌تواند با استفاده از آن توانایی‌های آکروبات را در برنامه خود به کار گیرد.

سامانه‌ی اطلاعاتی

سیستم اطلاعاتی یا سامانه اطلاعاتی یک پایگاه داده برای ذخیره، پردازش و تجزیه و تحلیل نتایج گزارش‌هایی است که به‌طور منظم درحال انجام است. به این دلیل سیستم‌های اطلاعاتی نه تنها به عنوان یک پایگاه داده، نرم‌افزار و سخت افزار بلکه به عنوان یک سیستم بزرگتر که به کمک و بهره‌گیری ازآن می‌توان مدیریت تمام امور دستی و دستگاهی و تفسیر سیستم‌های ارتباطی را به راحتی انجام داد، مطرح می‌شوند.

با این حال، این اصطلاح در معنای وسیع تر به هر وسیله‌ای که باعث برقراری ارتباط علمی بین افراد (به‌طور مثال از طریق ارتباط کلامی، سیستم‌های پانچ کارت و سیستم‌های تصادفی نوری ساده) می‌شود، اطلاق می‌شود. همچنین گاهی اوقات تنها برای اشاره به نرم‌افزار مورد استفاده برای اجرای یک پایگاه داده کامپیوتری یا برای اشاره به یک سیستم کامپیوتری استفاده می‌گردد. مجموعه سیستم‌های اطلاعاتی، برای مطالعه علمی، دقیق و واقعی شبکه‌های متشکل از سخت‌افزار و نرم‌افزار، که مردم و سازمان‌ها برای جمع‌آوری، فرایند ایجاد و توزیع اطلاعات از آن‌ها استفاده می‌کنند مورد استفاده قرار می‌گیرند. هدف هر سیستم ارتباطی حمایت از عملیات، مدیریت و تصمیم گیری و در معنای وسیع تر کاهش مدت استفاده می‌باشد؛ که می‌توان به فناوری اطلاعات و ارتباطاتی که یک سازمان، یا تکنولوژی که مردم در حمایت از فرایندهای کسب و کار انجام می‌دهند، اشاره کرد.

لایه‌ی کاربرد شبکه و وب

مدل OSI:

این مدل برگرفته از کلمه‌یOpen System Interconnection است و برای ارتباط بین دو کامپیوتر مبدأ و مقصد به کار می‌رود. این مدل در سال ۱۹۸۰ توسط سازمان ISO طراحی و پیاده‌سازی شده است و طبق سالیان متوالی تغییراتی روی آن صورت گرفته است، هرچند که همان ساختار اصلی خود را حفظ کرده است.

این مدل بر اساس یکی سری قراردادها با لایه­ ی مقابل خود در کامپیوتر دیگر ارتباط برقرار می‌کند و این کار باعث افزایش سرعت و امنیت در شبکه خواهد شد.

**لایه کاربرد شبکه در مدل OSI**  
لایه کاربرد (Application Layer) را می توان ملموس ترین لایه در مدل مرجع OSI نام برد. زیرا حتی نا آشنا ترین افراد هم در حوزه شبکه حداقل یک بار با این لایه کار کرده اند. بطور مثال، همه‌ی ما با مرورگر ها کار کرده ایم. لایه اپلیکیشن بالا ترین لایه در مدل OSI می باشد و رابطی است بین کاربر و سیستم برای ارائه سرویس به کاربر. این ارتباط می تواند بصورت Command Line و یا بصورت گرافیکی باشد. این لایه سرویس ها و پروتکل های مورد نیاز ما را تامین می کند و همچنین وظیفه ارتباط با لایه های زیرین خود را نیز دارد.  
 چندین سرویس را که این لایه به ما ارائه می دهد:  
   
HTTP  
این سرویس برای زمانی است که کاربر مرورگر خود را باز کرده و اسم سایت مورد نظر خود را در قسمت URL وارد می‌کند. در این زمان لایه اپلیکیشن درخواست کاربر را بصورت HTTP ارسال می کند و درواقع درخواست می کند که محتوا یا Content سایت مورد نظر را برای ما به نمایش بگذارد.

File Transfer Protocol) FTP  
این سرویس مخصوص انتقال فایل می باشد. بدین صورت که در محیط های سازمانی برای ذخیره سازی اطلاعات و یا دسترسی به اسناد از این سرویس استفاده می کنند.  
  
Remote Access  
این سرویس در واقع به کاربر این امکان را می دهد که از راه دور به سیستم خود متصل شود. از جمله نرم افزار هایی که این سرویس را ارائه می دهند نرم افزار Team Viewer را می‌توان نام برد.  
  
SMTP  
این سرویس مخصوص درخواست ارسال Email می باشد.  
  
این نکته را باید بدانیم که لایه اپلیکیشن در واقع پروتکل ها را ایجاد می کند. برای رفع نیاز های کاربر و شرکت های مختلف نرم افزار هایی را بر اساس این پروتکل ها ایجاد می کنند که همان سرویس را به کاربر بدهد.

معماری client/server

شبکه client / server

سرور یک برنامه (کامپیوتری Process) است که دارای اطلاعات است و برای دیگر برنامه های کامپیوتری سرویس و خدمات فراهم می کند.

کلاینت یک برنامه (کامپیوتری Process) است که نیاز به اطلاعات دارد و از سرویس ارایه شده توسط سرور استفاده می کند.

ارتباط بین دو پروسس کلاینت و سرور با درخواست از طرف کلاینت و ارایه پاسخ از طرف سرور انجام می شود.

سرور به دیگر کامپیوترهای شبکه سرویس می دهد.در واقع به تقاضای کامپیوترهای دیگر پاسخ مناسب می دهد که این پاسخ

می تواند اطلاعات مورد تقاضای آن ها ویا اجرای دستورالعمل باشد.

برای تحویل درخواست ( Request) یا پاسخ (Response) نیاز به آدرس دیگری به نام شماره پورت است

تا آن به یکی از پروسس های در حال اجرا با پروتکل خاص روی هاست مقصد تحویل داده شود.

ويژگی ها :

* در معماری فوق از سرويس دهند گان و سرويس گيرند گان با خصايص متفاوت استفاده می شود.
* اصل تقسيم کار دنبال و سرويس دهنده عمليات سنگين با پردازش بالا و سرويس گيرنده عمليات سبک را انجام خواهند داد.
* دو بخش متفاوت يک برنامه ، در جهت انجام عمليات با يکديگر تشريک مساعی می نمايند.
* سرويس گيرنده با ارسال درخواست و سرويس دهنده با پاسخ به درخواست جلوه ای از همياری در پردازش عمليات را بنمايش می گذارند.
* پلات فورم و سيستم های عامل سرويس دهنده و سرويس گيرنده می تواند متفاوت باشد.
* عملياتی را که يک برنامه انجام می دهد بين سرويس دهنده و سرويس گيرنده تقسيم می گردد.

مزايا :

* بهره گيری مناسب از پتانسيل های سخت افزاری موجود با توجه به اصل تقسيم عمليات ها
* بهينه سازی استفاده و بکارگيری منابع اشتراکی .
* بهينه سازی توانائی کاربران از بعد انجام فعاليت های متفاوت

معايب :

* عدم وجود امکانات لازم برای کپسوله نمودن سياست های راهبردی نرم افزار
* کاهش کارائی برنامه همزمان با افزايش تعداد کاربران همزمان
* بهبود عملکرد برنامه و يا اعمال اصلاحات مورد نظر همواره يکی از چالش های جدی است .

اتصالات کلاینت به سرور:

اتصال به سرور از طریق LAN

در اینگونه اتصالات کامپیوترها بوسیله کابل هایی به سرور متتصل میشوند.

خط تلفن:

کامپیوتر کلاینت بوسیله خط تلفن با یک سرور از راه دور میتواند ارتباط برقرار کند.

یک شبکه WAN:

در این مورد اتصالات مبتنی برTCP/IP است و در اینترنت انجام میشود.

یک شبکه بی سیم:

این گزینه که در سطح وسیع انجام نمیشود امکان اتصال کامپیوترها بوسیله امواج بی سیم را فراهم میکند.

انواع پردازش

۱ - پردازش های مبتنی بر ميزبان .

مدل فوق بمنزله يک مدل سرويس دهنده / سرويس گيرنده تلقی نشده و مشابه مدل MainFarme است .

۲ - پردازش های مبتنی بر سرويس دهنده .

در اين مدل سرويس دهنده تمامی پردازش های مربوطه را انجام و سرويس گيرنده مسئوليت ايجاد بخش رابط کاربر را برعهده خواهد د اشت.

۳ - پردازش های مبتنی بر سرويس گيرنده .

تمامی عمليات بر روی سرويس گيرنده انجام خواهد شد. عمليات مربوط به بررسی صحت داده ها و ساير عمليات مربوط به منطق بانک های اطلاعاتی بر روی سرويس دهنده انجام خواهد شد.

٤ - پردازش های مبتنی بر همياری .

در اين مدل سرويس دهنده و سرويس گيرنده بمنظور انجام يک فعاليت با يکديگر تشريک مساعی خواهند کرد.

شماره پورت port number

شماره پورت یک عدد 16بیتی است برای تعیین این که یک بسته اطلاعاتی در شبکه به چه برنامه ای در هاست مقصد تعلق دارد.

برای انتقال صفحات وب،پورت شماره 80 (HTTP)

برای انتقال فایل،پورت شماره 21 (FTP)

برای انتقال ایمیل پورت شماره 25 (SMTP)

**معماری توزیع شده client/server**

1. کلیه خصوصیات این نوع از معماری شبیه ThreeTier است.
2. انواع سرور در این معماری وجود دارد.
3. در یک معماری توزیع شده نقاط منطقی ممکن است ازتعدد بالایی برخوردار باشند.
4. از نقاط قوت معماری توزیع شده کاهش بار ترافیک در محیط هایی با پهنای باند کم میباشد.

توزیع شدگی:

مبتنی بر توزیع بار کاری در اعضای یک سیستم به منظور دستیابی به کارایی و بهره وری بالاتر.

معماری توزیع شده راه حل مناسبی برای طراحی ساخت سیستم هایی است که خصوصیات زیر راداشته باشند:

1. قدرت پردازش در حجم بالا
2. پهنای باند ارتباطی کم
3. تحمل خرابی
4. توان عملیاتی
5. فشار زمانی بالا
6. پشتیبانی ساده

انتخاب الگوی توزیع

برای انتخاب الگوی مناسب باید به معیارهای زیر توجه کنیم.

1. مقیاس پذیری
2. هزینه
3. پشتیبانی ساده
4. تحمل خرابی

**کلاینت و سرور مبتنی بر وب**

با رشد اينترنت(هم چنین اینترانت و اکسترانت)، به مرور توسعه نرم افزارها به صورت Web Based Application مورد توجه قرار گرفت

در این مدل دیگه نیازی به پیاده سازی سوکت پروگرمینگ نیست چون نرم افزار تحت وب ذاتا آن را در خودش دارد.در این روش تعداد کاربران نامحدود است.

بیشتر کاربران دارای ابزاری کلاینتی برای دسترسی به محتوای وبی هستند چون که سیستم عامل های امروزی شامل مرورگرهای وب نیز می باشند.

برنامه های کلاینت سروری تحت وب به صورت URL-Driven است.

FAT CLIENT

یک Fat Client با وجود اینکه به سرور متصل می شود ولی بیشتر پردازش ها در سمت کلاینت Client Side انجام می شود.

با Fat Client پردازش های زیادی از طریق شبکه انجام نمی شود و برای شبکه هایی با تراکم بالا و یا سرعت ارتباطی پایین بسیار مناسب است.

مزایا:

1. هزینه پایین تر برای راه اندازی سرور
2. کارایی بیشتر در نرم افزار های چندرسانه ای
3. بالا رفتن ظرفیت سرور برای پشتیبانی کردن از تعداد بیشتری کاربر

THIN CLIENT

اساساً Thin Client یک اپلیکیشن web based است و بیشتر پردازش ها در سمت سرورServer Side انجام می شود.

Thin clientجهت انجام بسياري از كارهاي خود به شدت وابسته به سرور است

مهم ترین مزیت Thin Client مدیریت‌ و نظارت آسان است.

معمولا نرم افزار Thin Client روی یک کامپیوتر با حداقل امکانات و اندازه کوچک و باریک نصب می شود.

نمونه هایی از معماری Client Server

اینترنت بستری مناسب برای پیاده سازی برنامه های توزیع شده است . وجود زیر ساخت های لازم در این زمینه شرایط مطلوبی را برای پیاده سازی برنامه های توزیع شده ، فراهم آورده است

دستیابی به یک وب سایت و یا ارسال و دریافت پیام های الکترونیکی نمونه هایی ا زبرنامه های توزیع شده بر روی بستر اینترنت می باشند.

برنامه های پست الکترونیکی نمونه ای مناسب در این زمینه می باشند

برنامه های Chat نمونه دیگری از برنامه های توزیع شده بر روی بستر اینترنت می باشند.

نرم‌افزار تحت وب

در نرم افزارهای تحت وب دیگر پردازش های مربوط به نرم افزار مورد نظر بر روی سیستم کاربر صورت نمی گیرد. کاربر یک درخواست به سیستم خدمات دهنده(سرور) ارسال می کند، سیستم خدمات دهنده بر روی بستر اینترنت قرار دارد و به عبارتی تحت وب است، سپس سرور پردازش های لازم را انجام داده و نتایج را برای کاربر ارسال می کند.

بدین شکل کاربر با ساده ترین و قدیمی ترین سیستم ها نیز می توانید به قوی ترین خدمات تحت وب و نرم افزارهای تحت وب دسترسی داشته باشید زیرا دیگر نیازی به یک پردازنده قدرتمند ندارید و سیستم کاربر فقط وظیفه نمایش اطلاعات را بر عهده دارد نه پردازش داده ها را.

از طرفی وقتی کاربر یک نرم افزار را روی رایانه شخصی خود نصب می‌کند برای دسترسی به این نرم افزار باید حتما از همان رایانه استفاده کنید ولی در نرم افزارهای تحت وب دیگر کاربر محدود به هیچ سیستم یا مکان خاصی نیستید. هر کجا که اینترنت باشد کاربر به خدمات نرم افزارهای تحت وب دسترسی خواهید داشت.

به طوری کلی می توان گفت تمام مزایای یک نرم افزار محلی در نرم افزارهای تحت وب وجود دارد ولی بسیاری از ویژگی های نرم افزارهای تحت وب در نرم افزارهای محلی قابل دسترس نیست. مزایای متعدد نرم افزارهای تحت وب دنیا را به سمت آنلاین شدن سوق می دهد، به طوری که در آینده ای نه چندان دور دیگر نیازی به سیستم های قدرتمند شخصی جهت پردازش و ذخیره سازی اطلاعات نداریم و وجود اینترنت پر سرعت، ما را از تمامی مزایای نرم افزارهای تحت وب بهره مند خواهد ساخت.

از جمله مهمترین مزایای نرم افزارهای تحت وب می توان به موارد زیر اشاره کرد:

* محدود نبودن به مکان خاص و در هر جایی که اینترنت باشد در دسترس خواهند بود
* محدود به یک سیستم خاص نیستند
* تمامی پردازش های پیچیده و سنگین بر روی سرور انجام شده و می توان با سیستمی ساده از خدمات نرم افزارهای تحت وب استفاده نمود.
* در داده های حساس می توان از امنیتی به مراتب بالا و قابل اطمینان برخوردار بود.

معماری نرم‌افزار و الگوی معماری

معماری لایه­ ای

لایه بندی یکی از رایج ترین تکنیک هایی است که طراحان نرم افزار برای بخش بندی و طراحی سیستم های پیچیده از آن استفاده می­کنند. به طوری که هر لایه‌­ای کار خودش را به صورت منسجم انجام می­دهد و از ریز کار لایه­ی دیگر باخبر نیست تنها از سرویس های لایه‌­ی پایین تر استفاده می­کند و یک سری سرویس را در اختیار لایه­‌ی بالاتر می‌­گذارد. همچنین لایه بندی باعث می­شود تا لایه ها کمترین وابستگی را به هم داشته باشند.

معماری 3 لایه

همزمان با اینکه معماری Client-Server در محبوبیت به سر می برد، دنیای شی گرایی نیز در حال رشد بود و و به دنبال راه حلی برای حل مشکلDomain Logic و یا به زبان ساده تفکیک منطق برنامه از سایر قسمت ها بود، در نهایت این تلاش ها منجر به معرفی معماری 3 لایه گردید. در این روش شما هر برنامه از سه لایه ی اصلی تشکیل می شود.

Presentation

بالاترین لایه Presentation نام دارد و در واقع لایه ایی است که کاربران آن را مشاهده می کنند. این لایه همان UI برنامه می باشد.

Domain Logic

در لایه دوم منطق برنامه قرار دارد، این لایه Domain Logic نام دارد.

لایه‌ای که منطق برنامه آنا قرار می‌گیرد، در واقع جایی که دستورات، شرطی ها و بررسی های یک فرایند نوشته می شود را Domain Logic می گویند.

Data Access

در نهایت در معماری سه لایه، لایه آخر مربوط به فعالیت های مرتبط با پایگاه داده می باشد و از همین رو نام این لایه را Data Access می نامند.

در این روش می توان به سادگی کدهای هر بخش را از یکدیگر جدا کرد و با استفاده از همین تفکیک پذیری ساده می توانید قسمت های مختلف یک پروژه را با تیم های تخصصی طراحی و پیاده سازی کرد، مثلا از بخش UI درخواست کرد بدون هیچ نگرانی UI صفحات مختلف و یا فرم های مختلف پروژه را طراحی کنند و سپس در بخش دیگر برنامه نویسان کدهای این صفحات و فرم ها را تکمیل کنند.

همچنین با جدا شدن لایه Data Access می توان کدهای کار با پایگاه های داده را در پروژه های مختلف به سادگی استفاده نمود.

همانطور که مشاهده می کنید حاصل این تفکیک ها به وجود آمدن لایه های تخصصی همچون Entity Framework است که تمامی مسائل مرتبط با پایگاه داده را مدیریت می کند.

همین جا می بایست به یک نکته اشاره کرد، در بسیاری از وب سایت ها سوالی مبنی بر بهترین معماری مطرح می شود. شاید بتوان این سوال را با سوالی مبنی بر لپ تاپ بهتر است و یا تبلت مقایسه کرد. هر دو این سوال ها وابسته به نیاز مصرف کننده دارند طبیعتا هر پروژه ای مانند هر ساختمانی نیاز به معماری خود دارد و نباید انتظار داشت معماری یک هتل چندین طبقه با معماری یک ساختمان کوچک یکسان باشد زیرا این معماری ها باعث افزایش هزینه و ایجاد مسائل پیچیده می شوند، در نهایت زمانی این هزینه ها مقرون به صرفه و ارزشمند خواهند بود که وابسته به هزینه های صورت گرفته سودآوری اتفاق بیافتد.

طبیعتا شالوده یک هتل با شالوده یک ساختمان کوچک هیچگاه قابل قیاس نیست، پس منطقی نیست برای پروژه های کوچک روش های پیچیده استفاده شود، اما نکته ای که همه در مورد آن اشتباه فکر می کنند این است که تصور می کنند پروژه های کوچک باید به صورت خیلی ساده و در واقع بدون هیچ معماری به صورت درهم کد نویسی و طراحی شود، این تفکر نیز طبیعتا اشتباه است باید در پروژه های کوچک نیز از معماری مناسبی استفاده کرد تا فردا بتوان با کمترین هزینه آن را رشد و توسعه داد زیرا تنها صنعتی که در دنیا هیچگاه به اتمام نمی رسد صنعت توسعه نرم افزار است و ممکن است همین پروژه کوچک تبدیل به یک پروژه بسیار عظیم شود.

معماری MVC و تفکیک دغدغه‌ها

عبارت MVC یا Model View Controller واژه‌ای است که در هنگام کار کردن با فریم ورک‌ها مورد استفاده قرار می‌گیرد.  
این عبارت شکلی خاص از ترکیب کدهای برنامه‌نویسی است. در این شکل از برنامه‌نویسی کدهای برنامه به سه بخش تقسیم می‌شوند. با تقسیم و طبقه بندی کدها، برنامه به صورت سه لایه مجزا از هم در می‌آید. برنامه‌نویسی لایه‌ای مزایای زیادی دارد که در محدوده‌ی این نوشته نمی‌گنجد. تنها به همین نکته بسنده می‌کنیم که در برنامه نویسی MVC بخش View وظیفه‌ی نمایش اطلاعات به کاربر را بر عهده دارد. بخش Controller دریافت اطلاعات از کاربر و پردازش آن را به انجام می‌رساند. و بخش Model کار ذخیره و بازیابی اطلاعات را عملی می‌کند.

فریم‌ورک

برنامه‌نویسان فارغ از زبان برنامه‌نویسی که با آن فعالیت می‌کنند با فریم ورک‌های برنامه‌نویسی نیز مواجهند. این فریم ورک‌ها مجموعه‌ای کدهای از پیش آماده، کتابخانه‌های برنامه‌نویسی و قوانین توسعه‌ی نرم‌افزار هستند.  
فریم‌ورک در اصل محیطی است که برای یک زبان برنامه ‎نویسی خاص ایجاد می‌ شود و باعث آسان‎تر شدن کار می‎ شود. بطور کلی می‌ توان فریم‎‌ ورک را مجموعه‌ای از ابزارها، کتابخانه‌ ها، قرارداد ها و بهترین شیوه‌ ها تعریف کرد که فریم‎ ورک‎ ها این امکان را به برنامه نویسان می‎ دهند تا با توابع و کلاس‎ های از پیش تعریف شده، ضمن ایجاد امنیت بیشتر برای برنامه‎ها، از کد کمتری در برنامه‎ های خود استفاده کنند. فریم ورک الزاماً چیزی ساخته ‎شده و بصورت پکیج در‎آمده برای عموم نیست؛ چرا که ممکن است یک فریم‎ ورک منحصراً برای شخص یا یک تیم طراحی شده باشد.

مدلسازی نرم‌افزار

در فرآیند تحلیل و طراحی سیستم‌های نرم‌افزاری به منظور ایجاد یک نمود تصویری قابل بحث و قابل توضیح، مهندسین نرم‌افزار اقدام به مدلسازی سیستم از جنبه‌های مختلف می‌کنند. برخی از این جنبه‌ها به طور کلی عبارتند از:

* مدلسازی داده‌ای
* مدلسازی ساختاری
* مدلسازی رفتاری

از جمله فواید مدلسازی می‌تواند به این موضوع اشاره کرد که توضیح ویژگی‌های محصول به کارفرما راحت‌تر شده و همچنین فرآیند عیب‌یابی می‌تواند پیش‌تر آغاز شود.

در این مستند از چند نمونه دیاگرام به منظور مدلسازی سامانه استفاده کرده ایم که در بخش مستندات فنی قابل مشاهده می‌باشد.

مستند‌های متداول پروژه‌های نرم‌افزاری

در پروژه‌های نرم‌افزاری مستندات مختلفی مورد نگارش قرار می‌گیرند که بسیاری از آن‌ها دقیقا مطابق با مستندات هر نوع پروژه‌ی دیگر هستند، از جمله «طرح پیشنهادی» یا همان پروپوزال. علاوه بر این مستندات پروژه، مستنداتی نیز در حوزه‌ی حرفه‌ای مهندسین نرم‌افزار قرار می‌گیرد که به طور مثال بر مسائل فنی یا کاربری سیستم نرم‌افزاری تمرکز دارند از جمله:

* سند نیازمندی‌ها که بیانگر ویژگی‌ها و قابلیت‌ها سیستم مورد توسعه است.
* سند طراحی/معماری که بیانگر ساختار و روابط اجزای سیستم در داخل و با محیط خارج آن است
* سند فنی که بیانگر کنوانسیون‌ها، قراردادها، الگوریتم‌ها و تشریح APIها می‌باشد
* سند راهنمای کاربری که توسط کاربران سامانه و تیم پشتیبانی مورد استفاده قرار می‌گیرد.

اصل DRY در توسعه‌ی نرم‌افزار

در حوزه ی توسعه ی نرم افزار، اصول و قواعد بسیاری وجود دارد که گاها یکی از دیگری مهم‌تر جلوه می‌کند اما یکی از اساسی‌ترین قواعد برنامه نویسی، قانون DRY است که مخفف واژگان Don`t Repeat Yourself به معنی«دوباره کاری نکن» است!  
  
این قانون توسط دو توسعه‌دهنده به نام های Andy Hunt و Dave Thomas ابداع شد که بسیاری از دیزاین پترن های معروف برنامه نویسی، ریشه در این قانون دارند  
  
برنامه نویسی که بتواند تشخیص دهد کدام بخش‌های کد اصطلاحاً Duplicate یا «مشابه» هستند و تمام تلاش خود را به کار بندد تا با استفاده از کلاس‌ها و فانکشن های مختلف، میزان استفاده از کدهای تکراری در سراسر برنامه را به حداقل برساند، در نهایت سورس کد تمیز تری تحویل خواهد داد که در آینده نگهداری چنین پروژه یی به مراتب راحت‌تر از سورس کدی است که پر است از کدهای مشابه!  
  
هرچه میزان کدهای دوپلیکیت در سورس کد بیشتر باشد، احتمال ایجاد باگ در آینده به مراتب بیشتر خواهد شد  
  
در فرایند توسعه ی نرم افزار، بخش‌های بسیاری از کد را می‌توان دید که تکراری هستند و قانون DRY دقیقاً برای چنین موقعیت هایی است.   
دوپلیکیت شدن در منطق نرم‌افزار می‌تواند به اشکال مختلفی جلوه کند که از جمله ی رایج ترین آن‌ها می‌توان به آبجکت هایی که از روی کلاس خاصی ساخته می‌شوند اشاره کرد و اینجا است که بسیاری از دیزاین پترن ها به داد توسعه دهندگان می آیند. در واقع، ابداع دیزاین پترن ها یا «الگوهای طراحی» جلوگیری از استفاده از کدهای مشابه است.   
  
علاوه بر دیزاین پترن ها، یکسری اصول کدنویسی که تحت عنوان SOLID شناخته می‌شوند نیز بر پایه ی اصل DRY هستند به طور مثال، حرف S در ابتدای SOLID به اصطلاح Single Responsibility اشاره دارد. به عبارت دیگر، هر کلاسی که در پروژه ی خود ایجاد می‌کنیم فقط و فقط باید مسئول یک کار باشد و در صورت نیاز به اعمال تغییرات در کلاس مد نظر، فقط و فقط باید یک دلیل برای ایجاد آن تغییر وجود داشته باشد نه اینکه کلاس مد نظر در جای جای نرم‌افزار برای کارهای مختلفی استفاده شده باشد و به هر دلیلی، نیاز به اعمال تغییرات در آن کلاس داشته باشیم.

حمله‌ی CSRF

حملات CSRF یا XSRF، در این حمله کاربر در صفحه مدیریت سایت خود لاگین است یا مهاجم از طریق یک لینک کاربر را مجبور به ورود به مدیریت سایت می‌کند بدون آنکه کاربر متوجه این موضوع باشد و در نهایت به صفحه مورد نظر دسترسی خواهد یافت. این حمله از طریق کشف آدرس هایی که محتوایی را تایید و ارسال میکنند اتفاق می افتد (مانند صفحات لاگین). کدهای اجرایی میتواند از قالب یک URL source در یک تصویر ذخیره شود بصورتی که کاربر از رخداد و اجرای URL آگاهی پیدا نکند. عموما تزریق از طریق image tag در HTML یا JavaScript انجام می شود در صورتی که در سایت بصورت غیر امن (http) متصل باشیم و یا کوکی ها روی مرورگر ذخیره شده باشند این حمله با احتمال موفقیت بالاتری اجرا خواهد شد.  
  
   
  
سرویس PayPal یک مرتبه با آسیب پذیری CSRF روبرو شده است. نمونه دیگر این مشکل در Google رخ داد، در حالی که مهاجم م‌یتوانست اطلاعات اکانت Google کاربر را کشف کند و به اکانت قربانی دسترسی یابد.  
عموما صفحات ثبت نام عمومی که با ارسال و دریافت اطلاعات طراحی شده اند بیشتر هدف این حمله قرار میگیرند و از متدهایی مانند GET یا POST میتواند استفاده کند.  
راهکار رایج برای مقابله این مشکل ایجاد یک رشته کد (توکن) بصورت تصادفی که هم برای کوکی استفاده میشود و هم در آدرس ایجاد شده بصورت پنهان وجود خواهد خواهد داشت. در هنگام ارسال فرم بررسی می شود آیا CSRFToken در فرم همان CSRFToken ذخیره شده در کوکی است یا خیر و در صورت یکسان بودن اجازه اجرا و ارسال اطلاعات و یا لاگین را خواهد داد.  
  
راهکارها:

* قرار گرفتن یک Token منحصر به فرد بصورت پنهان در بدنه درخواست HTTP ارسال میشود.
* استفاده از reauthenticate درخواست با استفاده از ابزاری مانند کد CAPTCHA
* استفاده از OWASP’s CSRF Guard

OWASP سازمان بین المللی جهت استاندارد کردن ایمن سازی در طراحی و پیاده سازی می باشد. (Open Web Application Security Project)

برنامه‌نویسی شیءگرا

«برنامه‌نویسی شی‌گرا» (Object-Oriented Programming) یا به اختصار OOP یک الگو یا شیوه تفکر در برنامه‌نویسی است که برگرفته از دنیای واقعی بوده و از دهه ۱۹۶۰ میلادی مطرح گشته است. به زبانی که از این الگو پشتیبانی کند، «زبان شی‌گرا» گفته می‌شود؛ Simula 67 و Smalltalk نخستین زبان‌های برنامه‌نویسی شی‌گرا هستند. ایده شی‌گرایی در پاسخ به برخی از نیازها که الگوهای موجود پاسخ‌گو آن‌ها نبودند به وجود آمد؛ نیازهایی مانند: توانایی حل تمامی مسائل پیچیده (Complex)، «پنهان‌سازی داده» (Data Hiding)، «قابلیت استفاده مجدد» (Reusability) بیشتر، وابستگی کمتر به توابع، انعطاف بالا و...  
  
رویکرد برنامه‌نویسی شی‌گرا «از پایین به بالا» (Bottom-Up) است؛ یعنی ابتدا واحدهایی کوچک از برنامه ایجاد می‌شوند و سپس با پیوند این واحدها، واحدهایی بزرگتر و در نهایت شکلی کامل از برنامه به وجود می‌آید. برنامه‌نویسی شی‌گرا در قالب دو مفهوم «کلاس» (Class) و «شی» (Object) ارایه می‌گردد. هر کلاس واحدی از برنامه است که تعدادی داده و عملیات‌ را در خود نگه‌داری می‌کند و هر شی نیز حالتی (State) مشخص از یک کلاس می‌باشد.  
  
در برنامه‌نویسی شی‌گرا، هر برنامه در قالب موجودیت‌های کوچکی که در واقع همان اشیا هستند و با یکدیگر تعامل دارند در نظر گرفته می‌شود. برای داشتن این اشیا می‌بایست ابتدا کلاس‌های برنامه را تعریف نماییم؛ هر کلاس «رفتار» (Behavior) و «صفات» (Attributes) اشیایی که قرار است از آن ایجاد شوند را تعریف می‌کند. از یک کلاس می‌توان هر تعداد که بخواهیم شی ایجاد نماییم. هر شی بیانگر یک «حالت» یا یک «نمونه» (Instance) از کلاس خود است.

نگاشت داده‌ها (Data Mapper و ORM)

ORM (الگوی طراحی یا تهاجمی ضد الگو)  
مدیریت دیتا (اطلاعات) در برنامه های شیءگرا امری بسیار ضروری و مهم است . اینکه دریک برنامه ما چگونه بتوانیم اطلاعاتی را که کاربر و یا خودمان در حین اجرای برنامه و هنگامی که برنامه از حالت اجرا خارج شده است ، حفظ ، مدیریت و تغییر دهیم بسیار حیاتی است.  
  
ORM یا Obejct Relational Mapper مجموعه ای از تکنیک های برنامه نویسی برای تبدیل داده ها بین سیستم های غیر شی گرا با محیط های برنامه نویسی شی گراست. به عبارت دیگر ، برای تطبیق بانک های اطلاعاتی رابطه ای (مثل SQL Server, MySQL و …) با زبان های برنامه نویسی شی گرا (مثل سی شارپ، جاوا و …) و استفاده از امکاناتی همچون Inheritance از ORM ها استفاده می کنیم. در واقع با استفاده از ORM ها و استفاده از ابزارهایی که ORM در اختیار توسعه‌دهنده قرار می دهد، توسعه‌دهنده ساختار رابطه ای بانک اطلاعاتی خود را به ساختار شی گرا برنامه نویسی خود متصل می نماید و به این ترتیب پیچیدگی های مربوط به ذخیره سازی ، بازیابی و … در بانک اطلاعاتی را به عهده ORM می سپارید.  
در پیاده سازی سیستم های بزرگ نرم افزاری معمولا بخشی از انرژی و وقت تیم به پیاده سازی لایه داده سیستم (DAL) که وظیفه اتصال به بانک اطلاعاتی و ذخیره و بازیابی داده از آن را دارد صرف می شود که با استفاده از یک ORM بخش قابل توجهی از این انرژی را می توان صرف صرفه جویی نمود.  
ORM محدود به هیچ زبان خاصی نیست و تنها یک تکنیک و سیستم است که می تواند در زبانهای مختلف پیاده سازی شود. در حقیقت ” یک شیء مجازی از دیتابیس” داریم که می تواند در زبانهای برنامه نویسی استفاده شود و خروجی دیگر یک آرایه نیست. تعداد این ORM ها خیلی زیاد است و هرکدام تعدادی پایگاه داده رو پشتیبانی میکنند. هر دو بسته “تجاری و رایگان” این تکنولوژی وجود دارد.  
اولین فریم ورک مناسب و خوبی که برای ORM تولید شد، EOF نام گرفت (Enterprise Objects Framework) که البته خود نیز دارای مشکلاتی بود . در واقع Nhibernate ورژنOpen source و رایگان Hibernate (فریم ورک پیاده سازی ORM در Java) در دات نت است.  
  
مزایا  
 الف) یک ابزار ORM زمان تحویل پروژه را کاهش می دهد  
ب) یک ابزار ORM کدی با طراحی بهتر را تولید می کند  
ج) استفاده از یک ابزار ORM ، کار برنامه نویسی را ساده تر می کند

معرفی تکنولوژی‌های به کار رفته در توسعه‌ی سیستم

در این بخش با رویکردی عملی به معرفی ابزار و عناصری می‌پردازیم که به کمک آن‌ها و به کارگیری مفاهیم مطرح شده در بخش قبل سامانه را پیاده‌سازی کرده ایم.

ابزار کنترل نسخه‌ی کد منبع گیت

گیت یک نرم‌افزار کنترل نسخه و از مدل نرم‌افزارهای آزاد و متن‌باز برای بازنگری کد منبع توزیع شده و مدیریت منبع کد است که برای دنبال کردن تغییر فایلهای کامپیوتری و دنبال کردن کردن کارهای انجام شده روی آن‌ها توسط افراد مختلف است. هدف اولیه این نرم‌افزار برای استفاده در پروژه‌های نرم‌افزاری بوده‌است ولی می‌توان از آن تنها برای دنبال کردن تغییر فایل‌ها هم استفاده کرد.

گیت ابتدا برای توسعهٔ لینوکس توسط لینوس تروالدز به وجود آمد و اکنون پروژه‌های فراوانی از آن الهام گرفته‌اند. هر دایرکتوری کاری در گیت یک مخزن کامل با تاریخچهٔ کامل تغییرها و قابلیت بازنگری آن‌ها است و برای کار با آن نیازی به دسترسی به شبکه یا سرور مرکزی وجود ندارد. گیت یک نرم‌افزار آزاد است که تحت عنوان جی‌پی‌ال نسخه ۲ توزیع شده‌است.

سیستم عامل گنو/لینوکس

یک سیستم‌عامل شبه یونیکس است که بخش عمدهٔ آن سازگار با استاندارد پازیکس است. از دید فنی لینوکس تنها نامی است برای هستهٔ سیستم‌عامل و نه کل آن. هسته‌ای که نخستین بار در سال ۱۹۹۱ میلادی توسط لینوس توروالدز منتشر شد. اما به‌طور معمول این نام به سیستم‌عامل‌هایی که ازترکیب‌بندی لینوکس (به عنوان هسته سیستم‌عامل) با نرم‌افزارهای آزاد و متن‌باز بدست می‌آیند اطلاق می‌شود. در حالی که بنیاد نرم‌افزار آزاد سیستم‌عامل حاصل از این ترکیب‌بندی را «گنو/لینوکس» می‌نامد اما عموماً آن را لینوکس می‌خوانند. این دوگانگی در نام نهادن منجر به بحث نامگذاری گنو/لینوکس شده‌است.

محیط توسعه (و حتی ترجیحا عملیات) در این پروژه سیستم عامل گنو/لینوکس می‌باشد که در این مورد به طور خاص از توزیع Ubuntu 18.04 Bionic Beaver استفاده شده است.

زبان پایتون

پایتون از جمله زبان‌های برنامه‌نویسی قدرتمندی است که در زمینه علم داده‌ها، یادگیری ماشینی، خودکارسازی سامانه‌ها، توسعه وب، واسط‌های برنامه‌نویسی و... به کار گرفته می‌شود. شاید بتوانیم این ‌گونه بیان کنیم که پایتون در مقایسه با زبان‌های بزرگ یک زبان نسبتاً جدید به شمار می‌رود. این زبان برنامه‌نویسی در سال 1991 به دنیای برنامه‌نویسی وارد شد و به‌منظور پر کردن شکاف‌های موجود در دنیای برنامه‌نویسی و ارائه راهکاری به‌منظور نوشتن اسکریپت‌هایی که فرآیند انجام یکسری از کارهای روتین‌ خسته‌کننده را به ‌طور خودکار اجرا کنند یا ساخت یک نمونه اولیه از برنامه‌های کاربردی که در یک یا چند زبان دیگر پیاده‌سازی شوند، مورد استفاده قرار گرفت. با این حال در چند سال گذشته، پایتون به یکی از ابزارهای تراز اول در زمینه توسعه برنامه‌های کاربردی، مدیریت زیرساخت‌ها و تحلیل داده‌ها تبدیل شده است. امروزه پایتون در زمینه توسعه برنامه‌های کاربردی تحت وب و مدیریت سیستم‌ها و تجزیه و تحلیل بزرگ داده‌ها که رشد انفجاری به خود گرفته‌اند و همچنین هوش مصنوعی به یکی از بازیگران اصلی دنیای فناوری تبدیل شده است. پایتون این موفقیت چشم‌گیر و کاربرد گسترده را مدیون یکسری ویژگی‌های ارزشمندی است که هم در اختیار توسعه‌دهندگان حرفه‌ای و هم در اختیار توسعه‌دهندگان تازه‌کار قرار داده است. اساسی‌ترین کاربرد پایتون در ارتباط با اسکریپت‌نویسی و خودکارسازی است. پایتون تنها یک جایگزین برای اسکرپیت‌های شل یا فایل‌های دسته‌ای نیست، به‌واسطه آنکه از پایتون برای تعامل خودکار با مرورگرهای وب، برنامه‌های گرافیکی و پیکربندی سیستم‌ها از طریق ابزارهایی همچون Salt و Ansible می‌توان استفاده کرد.در حال حاضر، دو نسخه از زبان پایتون در اختیار برنامه‌نویسان قرار دارد. هریک از این دو نسخه ویژگی‌های خاص خود را دارند. پایتون نسخه 2.X نسخه قدیمی‌تر است و پشتیبانی رسمی و عرضه به‌روزرسانی‌ها برای این نسخه تا سال 2020 ادامه خواهد داشت. حتی این احتمال وجود دارد که پس از این تاریخ نیز به‌صورت غیررسمی از این نسخه پشتیبانی به عمل آید.جدیدترین نسخه‌ی این زبان نسخه‌ی ۳/۷ می‌باشد.

**دستور زبان پایتون:**

آشنایی با انواع دستورات در زبان برنامه نویسی پایتون  
در یک تقسیم بندی می توان دستورهای پایتون را در دو دسته ی دستورهای ساده (Simple Statements) و دستورهای مرکب (Compound Statements) قرار داد که به اختصار به بررسی این دو گروه می‌پردازیم:

دستورهای ساده:

تا به حال تمام چیزهایی که در زبان پایتون آموخته ایم را در قالب دستورات ساده ای در اختیار مفسر قرار داده ایم تا آن ها را اجرا کند. به طور مثال در دستور زیر:

|  |
| --- |
| >>> num1 = 4 + 5 >>> message = """Welcome to SokanAcademy.com""" >>> num1 += 8 >>> 2 + 3 >= 6 False |

همان طور که می بینیم بعضی از این دستورها خروجی خاصی ندارند، مثل سه دستور اول و در حقیقت با وارد کردن آن ها مفسر پایتون در پشت صحنه عملیات انتساب را انجام می دهد و ما بعداً می توانیم از نتایج این عملیات استفاده کنیم؛ مثلاً متغیر message که در این عملیات در حافظه ایجاد و به آبجکتی از نوع استرینگ "Welcome to SokanAcademy.com" منتسب شده است را چاپ کنیم. این در حالی است که دستور چهارم یک خروجی قابل مشاهده به ما نشان می دهد. همان طور که در این مثال ها می بینید هر یک از این دستورها را به صورت مستقل از هم در یک خط مثل دستور اول، یا در بیش از یک خط مثل دستور دوم نوشته ایم و اجرا می کنیم. در زبان پایتون اصطلاحاً به هر یک از این دستورهای ساده، یک Logical Line یا «سطر منطقی» گفته می شود.   
  
به عنوان مثالی دیگر دستورات ساده ی زیر را در نظر بگیرید:

|  |
| --- |
| >>> message = "Welcome to SyntaxError: EOL while scanning string literal >>> message = "Welcome to\ SokanAcademy.com" >>> message = "Welcome to \ # This is a line continuation character. SyntaxError: EOL while scanning string literal |

همان طور که در دستور اول می بینید، وقتی بدون استفاده از کاراکتر \ دکمه ی اینتر را می زنیم تا ادامه ی دستور را در خط بعدی بنویسیم، مفسر پایتون بلافاصله اعلام خطای سینتکسی می کند؛ با این حال زمانی که از کاراکتر \ استفاده می کنیم به راحتی می توانیم به سطر بعدی برویم و ادامه ی دستور خود را وارد کنیم. در دستور سوم هم می بینید که بعد از قرار دادن کاراکتر \ نمی توان آن خط را ادامه داد و توضیح در آن وارد کرد، هرچند که مفسر توضیحات را نادیده می گیرد.  
  
گرچه اکیدا توصیه می شود که دستورات ساده و مستقل از هم در خطوط جداگانه وارد شوند، با این حال برای نوشتن آن ها در یک خط کافی است بین آن ها کاراکتر ; را قرار دهیم..  
  
در این صورت فراموش کردن درج کاراکتر ; (سمی کالن) بین دستورات منجر به بروز خطای سینتکسی می شود.

* دستورهای مرکب:

دستورهای مرکب اجرای سایر دستورها را کنترل می کنند یا بر روند اجرای آن ها تأثیر می گذارند. به طور مثال دستورهای شرطی در این گروه جای می گیرند که در ساده ترین حالت ممکن در صورت برقراری یک شرط خاص مفسر یک سری از دستورات را اجرا می کند، و در صورتی که شرط برقرار نباشد آن دستورات را نادیده می گیرد و اجرا نمی کند  
  
یک دستور مرکب از یک یا چند Clause (کلاز یا بند) تشکیل می شود. یک بند از یک سر بند یا Header و یک بدنه یا Suite تشکیل شده است. هر سر بند با یک کلمه ی کلیدی یکتا آغاز می شود و با کاراکتر دو نقطه : پایان می یابد. پیش از این لیستی از کلمات کلیدی پایتون را به شما معرفی کردیم و گفتیم این کلمات شناسه هایی هستند که به صورت پیش فرض تعریف شده اند و معنای خاصی را به مفسر پایتون می رسانند. مثال هایی از تعریف هدر برای یک بند به صورت زیر است:

|  |
| --- |
| if x > y: def move(): |

این هدرها با کلمات کلیدی if و def آغاز شده اند و با : پایان می یابند بدنه، گروهی از دستورات هستند که اجرای آن ها در یک بند کنترل می شود. دستورات بدنه ی یک بند را می توان به دو صورت نوشت:  
  
1- تمام دستورات بدنه را در همان خطی که سربند را وارد کردیم بنویسیم و در صورتی که چند دستور وجود داشته باشد آن ها را با سمی کالن ; از هم جدا کنیم. فرم کلی چنین بندی به صورت زیر خواهد بود:

|  |
| --- |
| Clause Header : Statement1 ; Statement2 ; ... |

به طور مثال:

|  |
| --- |
| if x > y: x = 0; y += 1; z = x+y |

همان طور که می بینید در مثال بالا بدنه از سه دستور ساده تشکیل شده است که در همان خطی که هدر قرار دارد نوشته شده اند و با سمی کالن از هم جدا شده اند.  
  
2- دستورات بدنه را در سطرهای مجزا که در یک بلوک کد مجزا از بلوک سطر سربند قرار دارند بنویسیم. بلوک بندی دستورات در بسیاری از زبان های برنامه نویسی هم چون جاوا و سی با استفاده از آکولادهای باز و بسته {} انجام می شود؛ حال این که در پایتون برای بلوک بندی کدها از تورفتگی یا Indentation استفاده می شود. تورفتگی با استفاده از فشردن کلیدهای Space یا Tab و ایجاد فاصله از آغاز سطر مشخص می شود.  
  
استاندارد تورفتگی در پایتون مقداری برابر با فضای خالی ایجاد شده بواسطه ی چهار بار فشردن کلید Space یا یک بار فشردن کلید Tab است. البته در زمان نیاز، IDLE به صورت خودکار تورفتگی هر بلوک را ایجاد می کند.  
مثالی از یک دستور مرکب که به این فرم نوشته شده است:

|  |
| --- |
| if x > y:  x = 0  y += 1  z = x+y >>> |

در حالت تعاملی IDLE زمانی که تمام دستورات بدنه را وارد کردیم کافی است دو بار کلید اینتر را فشار دهیم تا مفسر بلوک بدنه را ببندد و دستور مرکب را اجرا کند. اما در حالت اسکریپتی کافی است خودمان اشاره گر را جا به جا کنیم تا به بلوکی که هدر در آن قرار دارد برگردیم و ادامه ی دستورها را وارد کنیم:

|  |
| --- |
| if x > y:  x = 0  y += 1  z = x + y age = 5 |

بدنه ی دستور مرکب if با دستور z = x + y پایان یافته است و دستور بعدی از ابتدای سطر بعد آغاز شده است و تورفتگی ندارد. در صورتی که بدنه ی یک بند را به فرم دوم بنویسیم می توانیم از دستورات مرکب تو در تو استفاده کنیم. به طور مثال در قطعه کد زیر می بینید که دستور if دوم در دستور if اول آمده است:

|  |
| --- |
| if x > y:  if z > x:  print("z is the greatest.")  print("x is greater than y.") print("We found the greatest.") |

زبان جاوااسکریپت

یکی از مهم‌ترین زبان‌های برنامه‌نویسی حوزه‌ی وب محسوب می‌شود. جاوا اسکریپت اولین بار با اسم LiveScript و توسط شرکت نت اسکیپ معرفی شد و بعدها به جاوا اسکریپت تغییر نام داد. جاوا اسکریپت نه از نظر ساختار و نه از نظر مفاهیم شباهتی به زبان جاوا ندارد و این تشابه اسمی، در حد همان کلمات و نام و نشان باقی مانده است.

استاندارد جاوااسکریپت اکمااسکریپت (ECMAScript) است که از سال ۲۰۱۲ تمامی مرورگر‌های مدرن استاندارد اکمااسکریپت نسخه ۵.۱ را به صورت کامل پشتیبانی می‌کنند، همچنین مرورگر‌های قدیمی‌تر نسخه ۳ از اکمااسکریپت را پشتیبانی می‌کنند. از ماه June سال ۲۰۱۵ اکمااسکریپت ۶ (ES6) یا همان اکمااسکریپت ۲۰۱۵ (ES2015) مورد قبول واقع شده است  
  
 به‌طور طبیعی بعد از یادگیری HTML و CSS نوبت یادگیری جاوا اسکریپت می‌رسد. یادگیری جاوا اسکریپت چندان کار سختی نیست؛ ولی نکته مهم در رابطه با یادگیری این زبان، آن است که بر خلاف اچ‌تی‌ام‌ال یا سی‌اس‌اس که زبان‌های نشانه‌گذاری محسوب می‌شوند، جاوا اسکریپت یک زبان «برنامه نویسی» است

دستور زبان جاوا اسکریپت:

حساسیت به بزرگی و کوچکی کاراکترها

جاوااسکریپت زبانی حساس به بزرگی و کوچکی کاراکترهاست. حتماً لازم است که نام سازنده ( Constructor) را با حرف بزرگ و نام یک تابع را با یک حرف کوچک شروع کنیم.

فضاهای خالی(فاصله) و سمی کولن

بر خلاف زبان‌هایی مانند C زبان جاوااسکریپت حساس به فاصله‌است. با استفاده از سمی کالن گذاری می توان خطای ناشی از فاصله گذاری نامناسب را جبران کرد. نمونه‌هایی از خطاهای فاصله گذاری در ادامه رخ داده‌است:

|  |
| --- |
| return a + b;  // Returns undefined. Treated as: // return; // a + b; |

اما:

|  |
| --- |
| a = b + c (d + e).foo()  // Treated as: // a = b + c(d + e).foo(); |

توضیحات

قواعد توضیح گذاری در جاوااسکریپت عیناً مشابه زبان سی++ است:

|  |
| --- |
| // a short, one-line comment  /\* this is a long, multi-line comment  about my script. May it one day  be great. \*/  /\* Comments /\* may not be nested \*/ Syntax error \*/ |

متغیرها

شناساگرها(Identifiers) در زبان جاوااسکریپت دارای نوع خاصی نیستند و هرنوع مقداری می‌تواند داخل هر داده‌ای قرار بگیرد. تعریف دادهٔ جدید با استفاده از دستور var انجام می‌شود. می توان چندین متغیر را به یکباره تعریف کرد. اسم یک متغیر می‌تواند با یک حرف، با یک underline یا یک علامت دلار یا $ شروع شود. کاراکترهای دیگر می‌تواند اعداد ۰ تا ۹ باشد. با توجه به اینکه جاوااسکریپت حساس به بزرگی-کوچکی حروف است، استفاده از کاراکترهای بزرگ یا کوچک، متغیرهای جدیدی را ایجاد می‌کند. با شروه جاوااسکریپت نسخهٔ 1.5، ISO 8859-1 یا کاراکترهای یونی‌کد را می توان در ساختار متغیرها استفاده کرد.در پیاده سازی‌هایی از جاوااسکریپت از علامت @ می توان به عنوان یک شناساگر استاده کرد. در حالی که در برخی از ویرایش‌های جدید از آن چنین امکانی وجود ندارد. مشابه بسیاری از زبان‌های دیگر، متغیرهای این زبان در محدوده‌ای که تعریف شده‌اند دارای اعتبار هستند(یا به اصطلاح lexically scoped).

زمانی که جاوااسکریپت به دنبال شناسایی مقدار یک شناساگر است، به دنبال تعریف آن و مقدار دهی آن در ناحیهٔ مورد نظر می‌گردد. در صورتی که در آن ناحیه تعریف یا مقداردهی پیدا نکند، به ناحیهٔ بالاتر مراجعه می‌کند، تا اینکه به ناحیه می‌رسد که در آن متغیرهای جهانی تعریف می‌شوند. اگر تعریف متغیر پیدا نشود، جاوااسکریپت یک استثنای ( Exception) ReferenceError بر می گرداند. زمانی که یک شناساگر مقداردهی می‌شود، دوباره همین مراحل تکرار می‌شود تا نزدیک نرین متغیر پیدا شود و مقداردهی به آن انجام گیرد. در صورتی که متغیری پیدا نشود، متغیری متناسب با نوع مقدار در ناحیهٔ جهانی تعریف می‌شود. حواستان باشد که جاوااسکریپت در حالت strict در صورتی که متغیری تعریف نشده باشد، در صورت فراخوانی، متغیر جدید تعریف نمی‌کند.

نوع تعریف نشده

نوع داده‌ی تعریف نشده ( undefined) مربوط به متغیرهایی است که ایجاد شده اما مقداردهی نشده‌اند. این نوع همچنین در فراخوانی اشیایی که هنوز ایجاد نشده‌اند برگردانده می‌شود. در مورد نوع‌های بولین، مقدار متغیرهایی که تعریف نشده باشند،false برگردانده می‌شود. یکی از رفتارهای غیرمترقبه استفاده از نوع تعریف نشده، و عملکرد false آن در جایگاه‌های منطقی است.

|  |
| --- |
| var test; // variable declared but not defined,...  //... set to value of undefined var testObj = {}; alert(test); // test variable exists but value not...  //... defined, displays undefined alert(testObj.myProp); // testObj exists, property does not,...  //... displays undefined alert(undefined == null); // unenforced type during check, displays true alert(undefined === null); // enforce type during check, displays false |

باید توجه کرد نوع "تعریف نشده" به صورت یک نوع صریح کامل تعریف نشده‌است. لذا استفاده از (x == undefined) راه کاملاً درستی برای "تعریف نشده" بودن یک متغیر وجود نیست؛ چرا که در نسخه‌های قبل از ECMAScript 5 این مجاز است که متغیری به صورت var undefined = "I'm defined now"; تعریف شود. شیوه‌ای بهتر استفاده گردن از (typeof x == undefined') می‌باشد.

|  |
| --- |
| alert(null == undefined); // unenforced type during check, displays true alert(null === undefined); // enforce type during check, displays false alert(typeof null == object'); // true |

## فریم‌ورک جنگو

جنگو یک فریم ورک و چهار چوب سطح بالا به زبان پایتون برای وب نرم افزاری می باشد که باعث سهولت ایجاد وب سایت هایی داینامیک می شود . یک فریم ورک سطح بالا بسیاری از موارد برنامه نویسی را بصورت خودکار فراهم کرده و در اختیار برنامه نویس قرار می دهد . همچنین روش هایی میانبر و واسط برای اجرای اعمال مختلف را دارا می باشد . پس برنامه نویس نیازی به نوشتن کد هایی اضافی و مشخص را ندارد .  
  
یک وب سایت پویا وبسایتی است که شامل کد های ساده Html نبوده بلکه بصورت پویا و توسط نرم افزار (web application) تگ های اچ تی ام ال و در حقیقت صفحات سایت از روی کد ها و دستورات نوشته شده توسط برنامه نویس ایجاد می گردد . برای مثال ممکن است بر اساس درخواست کاربر اطلاعاتی به صورت داینامیک از پایگاه داده دریافت و نمایش داده شود .  
  
ویژگی های یک فریم ورک خوب عبارتند از :

مکانیسمی برای نگاشت URL درخواست شده به کدهایی که وظیفه پاسخ دادن به آن درخواست را دارند

به عبارت دیگر امکان تعیین اینکه کدام کد یا تابع مسئول جواب دادن به آدرس درخواست شده می باشد . برای مثال می توان از فریم ورک خواست تا برای آدرس‌هایی شبیه به /users/joe/ کدی را اجرا کند که اطلاعات کاربران را نمایش می دهد .

تسهیل نمایش ، اعتبار سنجی و نمایش مجدد فرم های html

فرم های اچ تی ام ال یکی از مهمترین روش ها برای دریافت اطلاعات از یک کاربر وب می باشد . پس یک فریم ورک خوب باید دارای روش هایی مناسب برای نمایش و کار با این فرم ها باشد . (برای مثال امکان مشخص کردن خطاهای رخ داده)

تبدیل اطلاعات وارد شده توسط کاربر به ساختاری مناسب برای مدیریت و استفاده

برای مثال تبدیل اطلاعات وارد شده در یک فرم اچ تی ام ال به ساختاری متناسب با زبان مورد استفاده

امکان جداسازی محتوا از ظاهر نمایشی

در اکثر زبانهای موجود برنامه نویسی کد های اچ تی ام ال و در حقیقت ظاهر یا template سایت با کد ها و محتوای سایت آمیخته می گردد که باعث سختی و ایجاد مشکلاتی هنگام تغییرات بعدی و نگهداری می گردد . با استفاده از این روش ظاهر نمایشی سایت بصورت جداگانه در فایلی خاص ذخیره می گردد . اکنون با اعمال تغییر در هر کدام نیازی به ویرایش دیگری نیست و این دو موجودیتی مستقل می باشد .

امکان درونی سازی آسان با لایه اطلاعات ذخیره شده (storage)

در حالت معمول هر برنامه نوشته شده با جنگو دارای ۳ لایه مهم می باشد . بخش نمایشی (template) ، محتوا یا کد های کنترلی و بخش ذخیره دائمی اطلاعات (برای مثال پایگاه داده) . البته بسته به نوع برنامه و خواست برنامه نویس می توان این لایه ها را ادغام کرده یا نادیده گرفت .

امکان استفاده موثر از سطح بالایی از تجرید و انتزاع

جنگو در موارد زیادی با استفاده از مفهوم انتزاع سهولت بسیاری را فراهم کرده است . برای مثال برای کار با تکنولوژی چون ftp یا http با یک مفهوم انتزاعی و سطح بالا روبرو خواهید بود که با استفاده از روابط و توابع متعدد برنامه نویسی را بسیار آسان و قدرتمند می کند . البته امکان نفوذ و استفاده از سطوح پایین تر نیز فراهم است .

حذف موارد و مشکلات معمول در برنامه نویسی وب

حذف و رفع موارد و مشکلات معمول در برنامه نویسی سمت وب . برای مثال حذف پسوند فایل از آدرس های وب (aspx. یا php.)

فریم ورک django تمامی موارد فوق را دارا می باشد . این فریم ورک با استفاده از زبان زیبا ، قدرتمند و سطح بالای پایتون نوشته شده است . برای ایجاد وب سایت برنامه نویس کدهای پایتونی می نویسد که امکان استفاده از ویژگی های پایتون و کتابخانه های این زبان را دارد .

## فریم‌ورک بوت‌استرپ

Bootstrap مجموعه ای از ابزارهای رایگان برای ایجاد صفحات وب و نرم افزارهای تحت وب است که شامل دستورات HTML، CSS و توابع جاوا اسکریپت جهت تولید و نمایش فرم ها، دکمه ها، تب ها، ستون ها و سایر المان های مورد نیاز طراحی وب می باشد.  
Bootstrap در ابتدا توسط مارک اتو و جاکوب تورنتون و در جهت ایجاد یک چارچوب ظاهری مشخص و یکسان در ابزارهای توییتر طراحی و نوشته شد. قبل از شروع این پروژه نمونه های زیادی با همین رویکرد ایجاد شده بود که همگی با سرنوشتی مشابه و عدم استقبال طراحان وب دنیا مواجه شده بودند. به دلیل وجود مشکلات اساسی در نمونه های دیگر، سازنده اصلی توییتر یا همان مارک اتو تصمیم به ساخت یک سیستم داخلی و قدرتمند برای خود با نام Bootstrap گرفت. در اگوست سال 2011، توییتر Bootstrap را به صورت متن باز و رایگان به عنوان یک فریم ورک طراحی وب سایت در اختیار دیگران قرارداد و در فوریه 2012 به محبوب ترین پروژه در سایت Github تبدیل شده بود.

بوت‌استرپ از یک سری ماژول‌های مختلف تشکیل‌شده است که توسعه سریع‌تر و آسان‌تر ظاهر وب‌سایت را ممکن می‌سازد.

یکی از مهم‌ترین ویژگی‌های آن امکان طراحی واکنش‌گرا (رسپانسیو) است که به‌آسانی وب‌سایت شمارا در همه دستگاه‌ها از موبایل، تبلت تا صفحات دسکتاپ به‌درستی نمایش می‌دهد.

این فریم ورک از به‌روز ترین زبان‌ها مانند Html۵ و Css۳ پشتیبانی می‌کند.

کدهای Bootstrap، از دو پیش پردازنده مهم css یعنی LESS و SASS استفاده می‌کند.

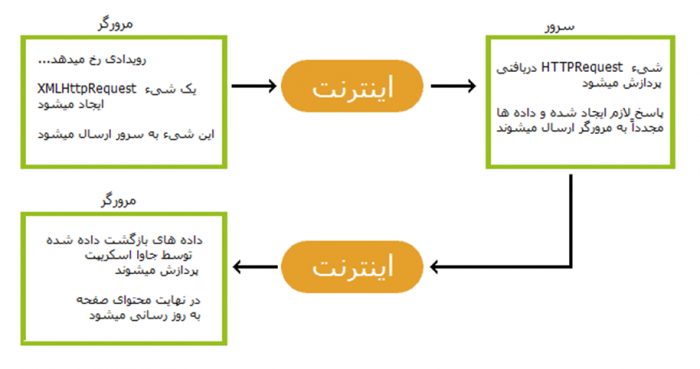
ساختار کلی بوت استرپ از یک سری فایل‌های استایل و اسکریپت تشکیل‌شده است.

در Bootstrap برای هر امکان کلاسی مشخص وجود دارد؛ مثلاً برای طراحی صفحه واکنشگرا و بخش‌بندی صفحه (Grid System ۴ ) کلاس lg-md-sm-xs تعریف‌شده است. این چهار کلاس مشخص‌کننده سایز باکس‌ها در دستگاه‌های مختلف است.

بوت‌استرپ ماژولار بوده و توسعه‌دهندگان می‌توانند فایل بوت‌استرپ را به انتخاب خودشان بسازند و حتی به‌وسیله دستورهای LESS تغییرات عمده‌تری انجام دهند.

## تکنولوژی AJAX

عبارت Ajax به طیف وسیعی از فن آوری های وب اشاره دارد که میتوانند برای پیاده سازی یک برنامه تحت وب بکار روند، فن آوریهایی که در پشت صحنه و در سرور فعالیت می کنند و در نتیجه با وضعیت جاری صفحه وب تداخلی ندارند. بطور کلی می‌توان گفت فن آوری‌های HTML (یا XHTML) و CSS مورد استفاده قرار میگیرند

به منظور نمایش محتوا فن‌آوری DOM ) Document Object Model یا مدل شیئی صفحه)، برای نمایش پویای داده ها و تعامل با آنها XML وJavaScript به منظور یکپارچه سازی این فن‌آوری ها البته در حال حاضر عمدتاً بجای XML (که فرمتی نسبتاً سنگین است) از JSON برای تبادل داده ها استفاده میشود و حتی امکان استفاده از HTML فرمت بندی شده یا متن ساده نیز به این منظور وجود دارد.  
  
 

تصویر 2نمود تصویری عملکرد AJAX

میتوانیم شکل بالا را در چند مرحله خلاصه کنیم:

* ابتدا یک رویداد client (یعنی یک رویداد در مرورگر) رخ میدهد
* یک شیء از نوع XMLHttpRequest ساخته میشود
* این شیء XMLHttpRequest با اطلاعات لازم پیکربندی میشود
* سپس این شیء یک تقاضای غیر همزمان به سرور وب ارسال میکند
* وب سرور نیز پس از انجام پردازش لازم، نتیجه را که حاوی شیء XML (یا JSON) است برگشت میدهد
* شیء XMLHttpRequest تابع callback (تابعی که پس از انتهای عملیات فراخوانی می‌شود) را صدا کرده و نتیجه حاصله (شیء XML) را پردازش می کند.
* در نهایت ساختار HTML صفحه به روز رسانی میشود

## پایگاه داده Sqlite

SQLite یک برنامه مدیریت بانک اطلاعاتی مبتنی بر زبان استاندارد SQL است.

بر خلاف مدل های مرسوم بانک اطلاعاتی که به صورت Client/Server هستند و نیاز به نصب و پیکربندی های خاص خود دارند، SQLite تنها یک برنامه مدیریت بانک اطلاعاتی مستقل است که نیازی به هیچ گونه نصب و پیکربندی ندارد.

بعبارتي SQLite یک پایگاه داده کوچک (حجمی در حدود 500 KB) که به زبان C در قالب یک کتابخانه (Library) نوشته شده است و از پایگاه داده های RDBMS به حساب می آید. این پایگاه داده بدون لیسانس (Public Domain) ، آزاد و open source منتشر می شود.

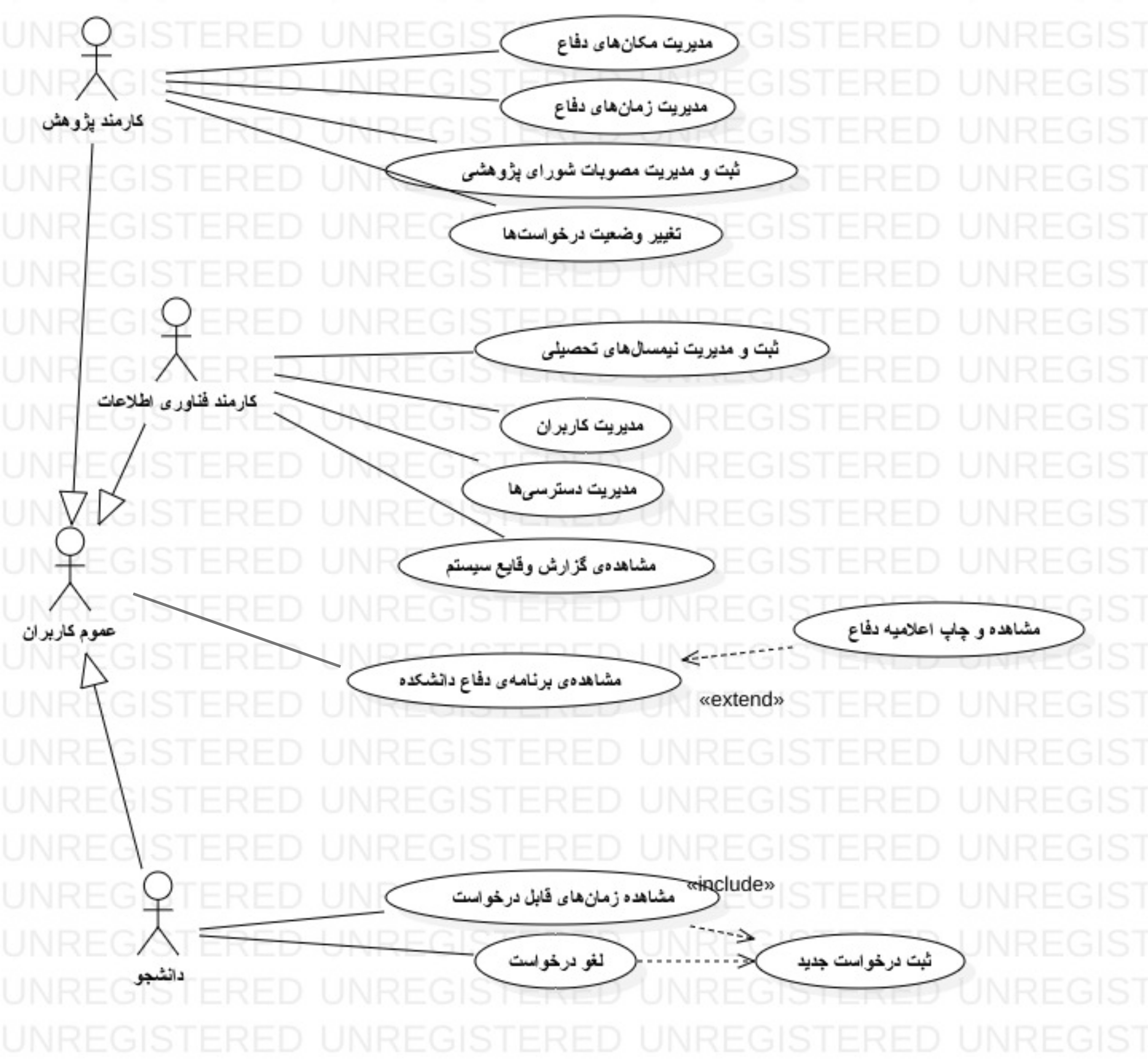
# مستندات فنی پروژه

در این فصل به ارائه‌ی مدل‌های بیانگر محصول در سطح کسب‌و‌کار مربوطه، در سطح پیاده‌سازی سیستم و همچنین تشریح عملکرد اجزای سیستم می‌پردازیم. برای این منظور از دیاگرام‌های متداول تحلیل و طراحی نرم‌افزار استفاده کرده و همچنین به توضیح قطعات کد سامانه می‌پردازیم.

## مدلسازی سیستم

در این بخش به ارائه‌ی نمودارهای سامانه و توضیح آن‌ها می‌پردازیم. این نمودار‌های شمای کلی سامانه را به ما نشان می‌دهند.

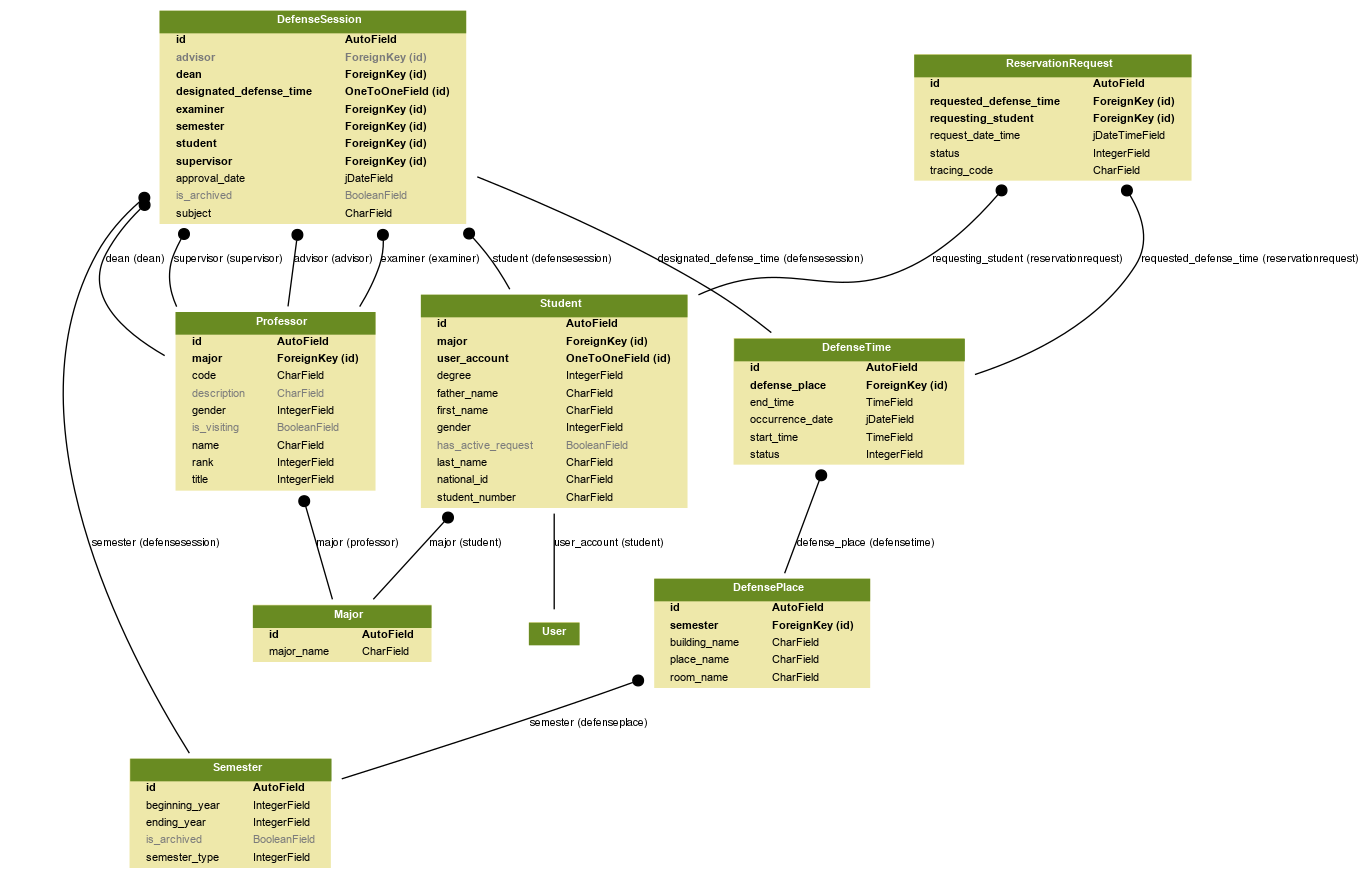
* **نمودار موارد کاربرد**



تصویر 3 نمودار موارد کاربرد (Use Case)

این نمودار مشخص می‌کند که چه قابلیت‌هایی می‌بایست در سامانه گنجانده شود و چه نوع کاربری باید بتواند به آن دسترسی داشته باشد. همچنین تقدم بعضی از عملیات‌ها نشان داده شده است. بطور مثال همه می‌توانند برنامه‌ی دفاع را مشاهده نمایند و البته به منظور چاپ اعلامیه‌ی مربوط به دفاع مورد نظر لازم است اول مشاهده‌ی برنامه‌ی دفاع (جدول زمان‌بندی شامل مشخصات تمام دفاعیه‌های در دستور کار) اتفاق بیفتد تا بتوان رکورد مورد نظر را چاپ کرد. همچنین ثبت و ربط کاربران تنها از عهده‌ فرد ذی‌صلاح در اداره‌ی فناوری اطلاعات بر می‌آید.

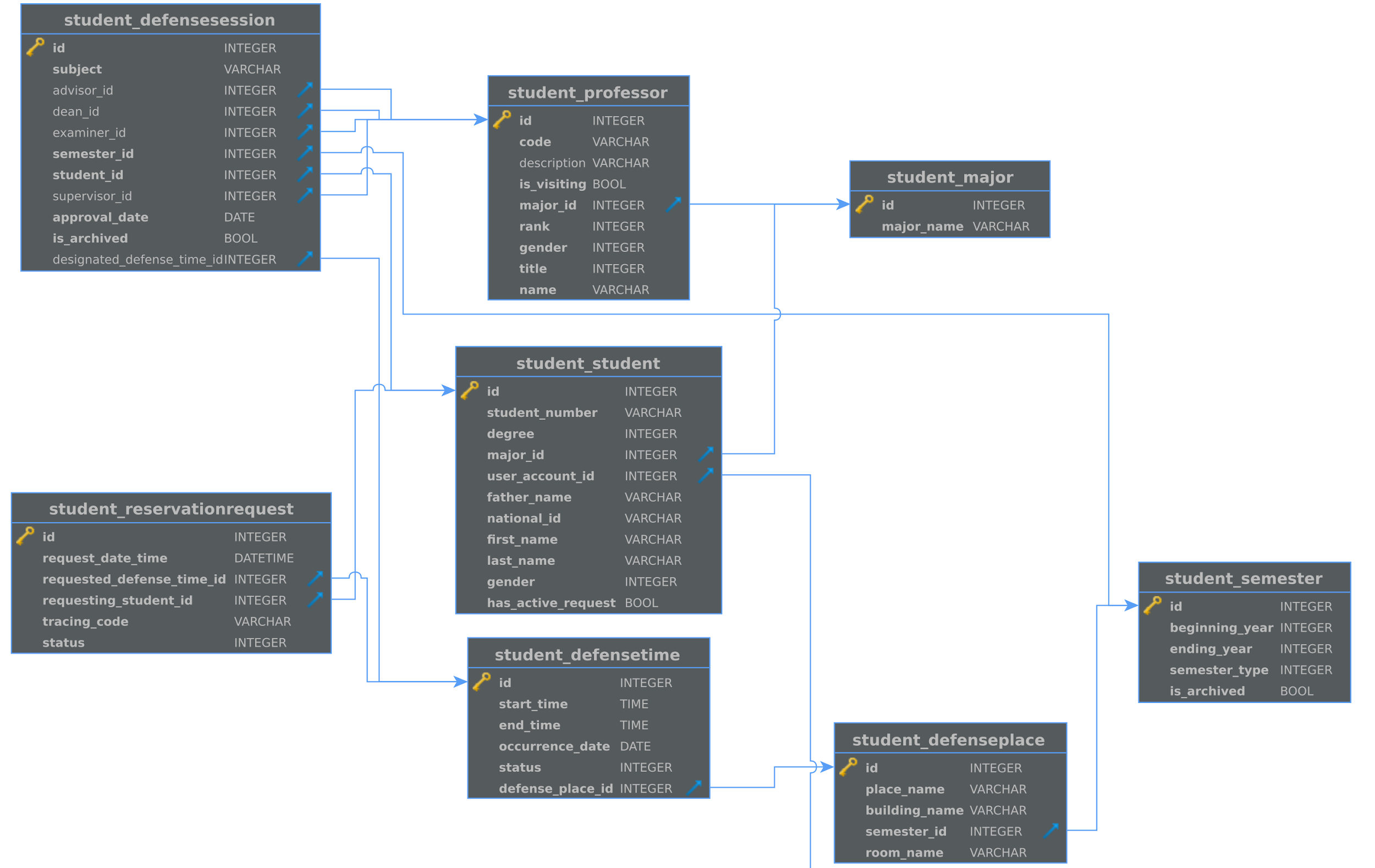
* **نمودار کلاس**



تصویر 4 نمودار کلاس

البته تصویر فوق دقیقا یک نمودار کلاس کامل نیست چرا که با وجود بهره‌گیری از شیءگرایی در سامانه، هیچ کلاسی به معنای کلاسیک آن موجود نیست و تقریبا تمام منطق‌های آن با استفاده از توابع پیاده شده. نمودار بالا نمایانگر کلاس‌های مدل سامانه است که برای نگاشت کردن جداول در ORM استفاده می‌شوند. در حقیقت این نمودار نمایش دیگری از نمودار ERD منتها در سطح اپلیکیشن است. تمام کلاس‌های فوق از کلاس (models.Model) ارث بری دارند که در این نمودار از نمایش آن صرف نظر شده است.

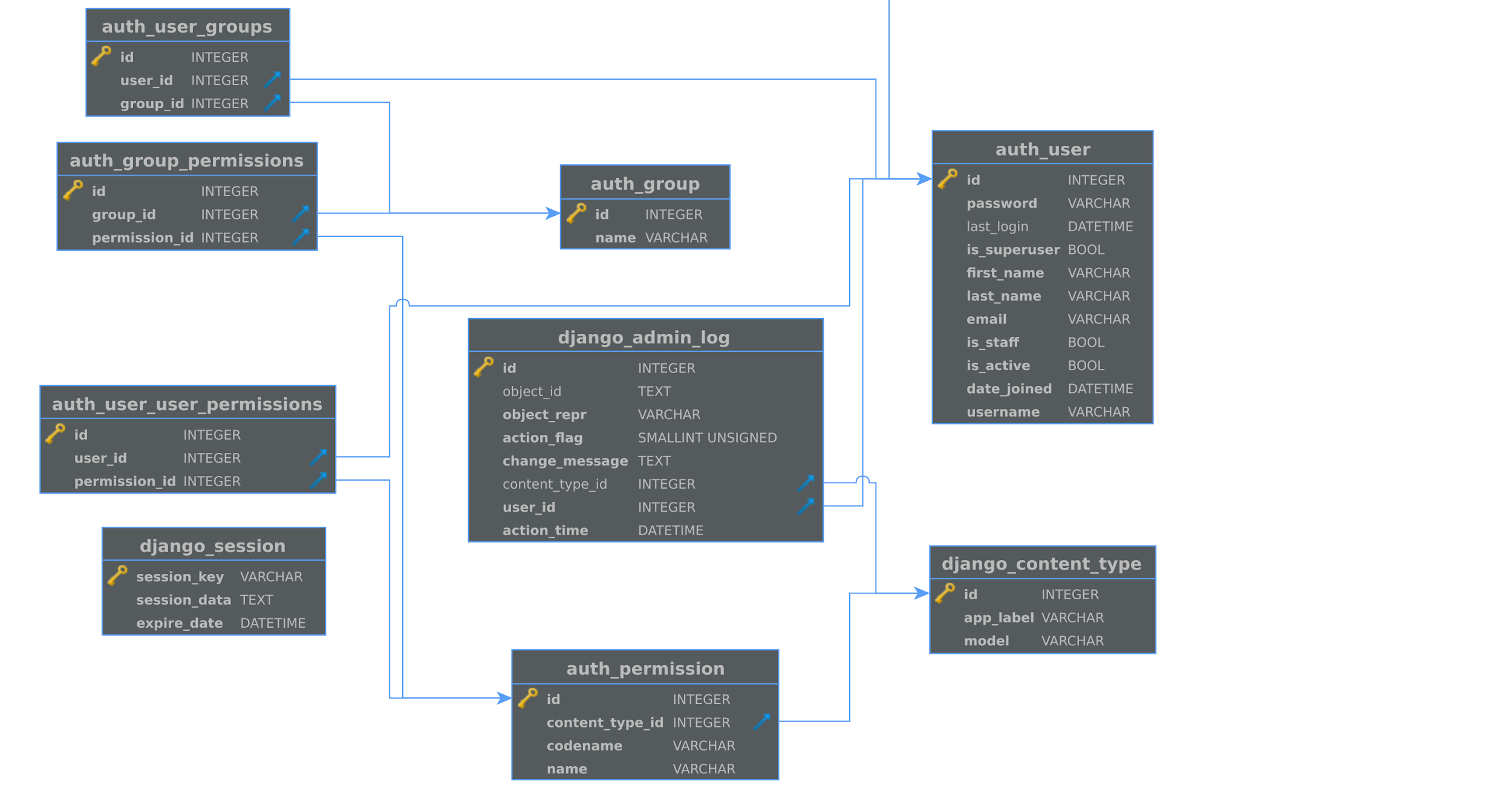
* **نمودار موجودیت-بستگی (اصلی)**



تصویر 5 دیاگرام موجودیت‌بستگی

این نمودار بیانگر شماتیک جداولی است که ما به واسطه‌ی مدل‌هایی که در قالب تعدادی کلاس در سطح اپلیکیشن به کمک نگاشتگر داده‌ها ساخته ایم و بستگی میان آن‌ها می‌باشد. اگرچه جداول پایگاه داده‌ی ما به این‌ها خلاصه نمی‌شود و شامل تعدادی جدول نیز هست که خود فریم‌ورک برای امور خودش و همچنین برای احراز هویت و تایید مجوز استفاده می‌کند. همانطور که دیده می‌شود این جداول همگی با پیشوند student نام‌گذاری شده اند. دلیل این پیشوند‌گذاری آن است که این موجودیت‌ها مربوط به اپ سامانه‌ی دانشجو می‌باشد که در پروژه از آن با نام «student» نام برده ایم. جدول student\_student نیز با موجودیت auth\_user که در تصویر زیر آمده بستگی یک به یک دارد.

* **نمودار موجودیت بستگی مربوط به فریم‌ورک**



تصویر 6 نمودار موجودیت بستگی جداول فریم‌ورک

در این نمودار جداول مربوط به فریم‌ورک جنگو و میان‌افزار auth که امور احراز هویت و اجازه‌سنجی (Authentication & Authorization) را بر عهده دارد قابل مشاهده است. این جداول بدون دخالت ما ایجاد شده، اگرچه می‌توانیم در آن‌ها دست ببریم و به دلخواه خود بهبودشان دهیم یا حتی بخشی از آن‌ها را از جمله auth از سامانه خارج نماییم.

تشریح اجزای سیستم

همانطور که در مقدمه گفته شد، این سامانه یک نرم‌افزار تحت وب است و از الگوی client/server تبعیت می‌کند و به همین دلیل از یک فریم‌ورک سمت سرور و یک فریم‌ورک سمت کلاینت استفاده می‌کند. در این بخش به تشریح ساختار فریم‌ورک سمت سرور (جنگو) میپردازیم.

در این فریم‌ورک یک سری قوانین و استاندارد وجود دارد که اگرچه در اکثر اوقات می‌توان آن‌ها را رعایت نکرد اما با رعایت آن‌ها کار بسیار آسان شده و سریع‌تر انجام می‌پذیرد.

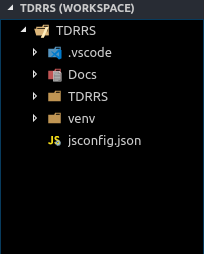
ساختار پروژه‌های جنگو

برای این که بتوانیم ساز و کار این سامانه را مخصوصا به منظور نگهداری و بارگذاری آن به خوبی درک کنیم، لازم است با ساختار فایل‌های آن آشنا شویم.

* **اجزای دایرکتوری پروژه‌ی جنگو**

در تصویر زیر دایرکتوری‌ها و فایل‌های گوناگونی به چشم می‌خورد. در محیط توسعه، بالاترین سطح یک پوشه‌ی اصلی داریم که نام آن اهمیتی ندارد (در این جا نام آن با نام پروژه یکسان است) و به طور کلی شامل دو پوشه‌ی مهم می‌شود: ۱- پوشه‌ی محتوی محیط مجازی (virtual environment) و ۲- پوشه‌ی محتوی فایل‌های سامانه. از این پس این پوشه را **پوشه‌ی والد** می‌نامیم. سایر فایل‌ها و پوشه‌هایی که در ذیل این دایرکتوری مشاهده می‌شود صرفا در محیط توسعه و فضای کار برنامه‌نویس مورد استفاده قرار گرفته اند و جزئی از پروژه نیستند.

پوشه‌ی محیط مجازی (در این جا venv) در حقیقت بخشی از پروژه نیست بلکه بخشی از محیط اجرای پروژه است و شامل مفسر پایتون، کتابخانه‌های لازم برای اجرای خود پایتون و همچنین پروژه و پکیج‌های مورد نیاز پروژه می‌باشد. از جمله ابزار‌هایی که حتما در این پوشه حضور دارد می‌توان به pip اشاره کرد. Pip مدیریت پکیج‌های پایتون را برعهده دارد.

****

تصویر 7 ساختار دایرکتوری محیط توسعه یا استقرار

پوشه‌ی دیگر که TDRRS نام دارد، دایرکتوری اصلی (ریشه) پروژه می‌باشد. این پوشه به طور متداول و متعارف شامل موارد زیر است:

1. دایرکتوری حاوی پیکربندی پروژه همنام با نام پروژه (در اینجا TDRRS)
2. دایرکتوری مربوط به هر کدام از اپ‌ها (زیر پروژه‌ها)
3. فایل اجرایی manage.py: این فایل ابزار اصلی ما در محیط توسعه است و اجرا، مهاجرت داده‌ها، ساخت اولین ابرکاربر سامانه و بسیاری دیگر از عملیات سامانه را این اسکریپت برای ما انجام می‌دهد و کافی است آن را با استفاده از مفسر پایتون اجرا کرده و دستورات را به شکل آرگومان به آن پاس بدهیم.

موارد متعارف دیگر که در این پروژه مشاهده می‌شود عبارتند از:

1. فایل requirements.txt: این فایل اطلاعات پکیج‌های مورد‌های مورد استفاده‌ی سامانه را در خود نگه می‌دارد تا در صورت تغییر محیط توسعه یا در هنگام استقرار سامانه به راحتی پکیج‌ها که در حقیقت dependency های سیستم‌ هستند نصب شده و راه‌اندازی شوند.
2. فایل .gitignore: این فایل شامل مشخصات فایل‌ها و دایرکتوری‌هایی می‌شود که git نباید تغییرات آن‌ها را دنبال کند.
3. فایل LICENSE: این فایل شامل مشخصات لیسانس تجاری و حقوق استفاده کننده از کد می‌باشد
4. فایل db.sqlite3: این فایل در حقیقت پایگاه داده‌ی sqlite3 است و تمام جداول و اطلاعات را در خود نگه می‌دارد.



تصویر 8 ساختار دایرکتوری اصلی پروژه

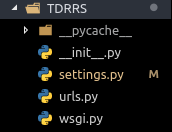
* **ساختار تنظیمات و پیکربندی پروژه جنگو**

همانطور که گفته شد دایرکتوری هم‌نام که در داخل پوشه‌ی پروژه قرار دارد دربردارنده‌ی پیکربندی سامانه است. محتویات آن در تصویر زیر مشهود است.

*تذکر: پوشه‌ی \_\_pycache\_\_ شامل کد میان‌سطح پایتون است که با یکبار اجرا به منظور اجرای سریع‌تر تولید می‌شود و هیچوقت کاری با آن نداریم. همچنین فایل \_\_init\_\_.py صرفا برای اعلان دایرکتوری حاوی آن به عنوان پکیج پایتون در پوشه حضور دارد.*

در این جا با ۳ فایل رو به رو هستیم:

1. فایل settings.py که شامل تمام تنظیمات حیاتی سامانه است
2. فایل urls.py که نقش url dispatcher را دارد.
3. فایل wsgi.py که برای عملیاتی کردن سامانه و اتصال آن به یک وب سرور عملیاتی استفاده می‌شود.



تصویر 9 ساختار پوشه‌ی مربوط به پیکربندی

حال بررسی فایل settings.py خالی از لطف نیست.

|  |
| --- |
| import os import locale  BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  LOGIN\_URL = '/student/login/'  SECRET\_KEY = 'clk-@)99e3j5v=85xl3%4t)(lsk6pdgnnditiy)g9qz34l9$e0'  DEBUG = True  ALLOWED\_HOSTS = []  PREREQ\_APPS = [  'bootstrap\_admin',  'django.contrib.admin',  'django.contrib.auth',  'django.contrib.contenttypes',  'django.contrib.sessions',  'django.contrib.messages',  'django.contrib.staticfiles',  'django\_jalali',  'django\_spaghetti',  'django\_extensions', ]  PROJECT\_APPS = [  'student',  'public',  'common' ]  INSTALLED\_APPS = PREREQ\_APPS + PROJECT\_APPS  MIDDLEWARE = [  'django.middleware.security.SecurityMiddleware',  'django.contrib.sessions.middleware.SessionMiddleware',  'django.middleware.common.CommonMiddleware',  'django.middleware.csrf.CsrfViewMiddleware',  'django.contrib.auth.middleware.AuthenticationMiddleware',  'django.contrib.messages.middleware.MessageMiddleware',  'django.middleware.clickjacking.XFrameOptionsMiddleware', ]  ROOT\_URLCONF = 'TDRRS.urls'  TEMPLATES = [  {  'BACKEND': 'django.template.backends.django.DjangoTemplates',  'APP\_DIRS': True,  'OPTIONS': {  'context\_processors': [  'django.template.context\_processors.debug',  'django.template.context\_processors.request',  'django.contrib.auth.context\_processors.auth',  'django.contrib.messages.context\_processors.messages',  ],  },  }, ]  WSGI\_APPLICATION = 'TDRRS.wsgi.application'   DATABASES = {  'default': {  'ENGINE': 'django.db.backends.sqlite3',  'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  } }   AUTH\_PASSWORD\_VALIDATORS = [  {  'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  },  {  'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  },  {  'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  },  {  'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  }, ]  LANGUAGE\_CODE = 'fa-ir' TIME\_ZONE = 'Asia/Tehran'  USE\_I18N = True  USE\_L10N = True  USE\_TZ = True  STATIC\_URL = '/static/' STATICFILES\_DIRS = [  os.path.join(BASE\_DIR, "static") ]  SPAGHETTI\_SAUCE = {  'apps': ['student'],  'show\_fields': False,  'exclude': {'auth': ['user']}, } |

در این فایل با یک سری متغیر روبه‌رو هستیم که مقادیر مهمی را در خود نگاه می‌دارند. در این جا می‌توان پکیج‌ها، اپ‌ها، زبان سامانه، ساختار ساعت سامانه، پایگاه‌داده‌ی مورد استفاده، وضعیت توسعه و دیباگ سامانه و یک سری مقادیر پیش‌فرض دیگر را اعلان کرد.

* **پیکربندی یک «اپ» در جنگو**

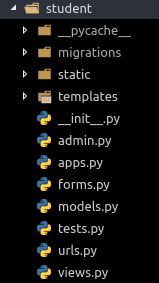
یک پروژه‌ی متعارف و مبتنی بر اصول در جنگو به این گونه است که باید به ازای هر دسته از قابلیت‌های خود یک اصطلاحا app داشته باشد. مثلا یک سامانه‌ی فروشگاهی برای کنترل احراز هویت یک اپ، برای بخش انبار یک اپ، برای مشاهده‌ی کالا یک اپ، برای ثبت سفارش و ویرایش آن یک اپ و برای پرداخت یک اپ دیگر داشته باشد.

پروژه‌ی TDRRS شامل ۳ دایرکتوری اپ با نام‌های «public»، «student» و «common» می‌باشد. البته common واقعا یک اپ نیست اما به عنوان یک اپ در سامانه معرفی شده و با آن به عنوان اپ برخورد می‌شود.

گفتنی است اپ student شامل منطق و مدل‌های مربوط به سامانه‌ی دانشجو می‌باشد و اپ public شامل منطق بخش عمومی سامانه که شامل مشاهده‌ی برنامه‌ی زمان‌بندی می‌باشد.

دایرکتوری یک اپ از فایل‌های متعددی تشکیل شده است که عبارتند از:

1. فایل admin.py که شامل یک سری پارامتر‌ها و تنظیمات رابط مدیریت می‌باشد و جداولی که باید در بخش مدیریت وبسایت در دسترس قرار بگیرد در این قسمت معرفی می‌کنیم.
2. فایل apps.py برای تنظیمات و پیکربندی خاص اپ می‌باشد.
3. فایل models.py که برای تعریف مدل‌ها (در حقیقت موجودیت‌هایی که قرار است در پایگاه داده‌ها به صورت جدول تعریف شوند) به صورت کلاس‌های پایتون استفاده می‌شود.
4. فایل tests.py برای نوشتن آزمون نرم‌افزار (یعنی تعدادی test case) خاص این اپ استفاده می‌شود و در متدولوژی TDD بسیار کاربرد دارد.
5. فایل urls.py در حقیقت url dispatcher خاص این اپ است و به این صورت کار می‌کند که url dispatcher اصلی آدرسی با ساختار /student/yyyy/zzzz را دریافت می‌کند و از آن جا که با student شروع شده ادامه‌ی پردازش آن را به url dispatcher اپ student منتقل می‌کند.
6. فایل views.py شامل توابع (یا کلاس‌هایی) می‌باشد که دربرگیرنده‌ی منطق سامانه هستند و درخواست‌های کاربر را بر اساس آدرس وارد شده پردازش می‌کند. در حقیقت عملکردی مانند controller دارند اما با انتزاع بیشتر.
7. پوشه‌ی static دربردارنده‌ی فایل‌های تصویری، کد js و استایل‌های css می‌باشد.
8. پوشه‌ی templates دربردارنده‌ی قالب‌های خام html است که بعدا توسط viewها پردازش شده و به کاربر ارائه می‌شود.



تصویر 10ساختار یک app

نمونه کد‌های سمت سرور

* قطعه کد یک view ساده:

|  |
| --- |
| from django.shortcuts import render  def get\_home(request):  return render(request,'public/index.html') |

این کد به این صورت کار می‌کند که اگر درخواستی به آن بیاید، فایل قالب index.html موجود در اپ public را پردازش کرده و برای کاربر ارسال می‌کند و مرورگر آن را نمایش می‌دهد.

* قطعه کد مربوط به pagination (یا صفحه‌بندی)

|  |
| --- |
| from django.shortcuts import render from student.models import \* from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger   def get\_schedule(request):  queried\_schedules=DefenseSession.objects.filter(is\_archived=False)  page = request.GET.get('page', 1)  paginator = Paginator(queried\_schedules, 5)  try:  page\_schedules = paginator.page(page)  except PageNotAnInteger:  page\_schedules = paginator.page(1)  except EmptyPage:  page\_schedules = paginator.page(paginator.num\_pages)  return render(request,'public/defense\_schedule.html',{'schedules':page\_schedules}) |

این کد به این صورت کار می‌کند که با توجه به درخواست آمده (آبجکت request) یک جستار در پایگاه داده انجام داده و با توجه به مقدار داخل query string صفحه‌‌بندی را با استفاده از paginator انجام می‌دهد و صفحه‌ی مورد نظر را به کاربر بازمی‌گرداند.

* قطعه کد مربوط به یک نمونه جستار (query)

|  |
| --- |
| def get\_announcement(request,dsid):  session=DefenseSession.objects.get(pk=dsid)  return render(request,'public/announcement.html',{'session':session}) |

این کد به این شکل کار می‌کند که علاوه بر شیء درخواست، یک مقدار عددی نیز از url dispatcher دریافت می‌کند. این مقدار عددی در حقیقت شناسه‌ی جلسه‌ی دفاع است. در خط دوم می‌بینیم که یک جستار با استفاده از ORM جنگو بر روی اشیاء مدل DefenseSession که نگاشتی بر جدول «جلسات دفاع» است می‌زند و رکوردی که کلید اصلی آن برابر با مقدار عددی dsid است را گرفته و در متغیری با نام session می‌ریزد. این متغیر به شکل یک شیء دیکشنری به تابع render پاس داده می‌شود تا در قالب announcement.html به منظور پردازش در دسترس قرار گیرد. بعدا این شیء در بدنه‌ی announcement به زبان DTL پردازش شده و محتویات فیلد‌های آن در صفحه‌ی پردازش شده مشاهده خواهد شد.

* محتویات فایل urls.py اصلی پروژه (URL Dispatcher اصلی)

|  |
| --- |
| from django.conf.urls import url from . import views  urlpatterns = [  url(r'^$', views.get\_dashboard, name='dashboard'),  url(r'^dashboard',views.get\_dashboard,name='dashboard'),  url(r'^login',views.do\_login, name='login'),  url(r'^defense-times',views.defense\_times, name='defense\_times'),   url(r'^logout$',views.do\_logout, name='logout'),  url(r'^change-password',views.do\_change\_password, name='change\_password'),   url(r'^cancel-reservation',views.do\_submit\_cancellation, name='cancel\_reservation'), ] |

این فایل url dispatcher اپ student است و آدرس‌هایی که با فرمت /student/yyy به سرور بیاید به این فایل منتقل می‌شوند. به طور مثال اگر مرورگر کاربر درخواستی به آدرس زیر بفرستد

127.0.0.1:8000/student/defense-times

خواهیم دید که که url چهارم انتخاب شده و کنترل را به views.defense\_times می‌دهد.

* یک نمونه کد migration

|  |
| --- |
| # -\*- coding: utf-8 -\*- # Generated by Django 1.11 on 2018-08-18 10:04 from \_\_future\_\_ import unicode\_literals  from django.db import migrations, models   class Migration(migrations.Migration):   dependencies = [  ('student', '0016\_auto\_20180809\_0054'),  ]   operations = [  migrations.AddField(  model\_name='semester',  name='is\_archived',  field=models.BooleanField(default=False, verbose\_name='بایگانی شده است؟'),  preserve\_default=False,  ),  ] |

این کد که توسط خود جنگو با دستور makemigrations تولید شده است به ORM می‌گوید به جدولی به نام semester ستونی به نام is\_archived از جنس boolean اضافه کند و به طور پیش‌فرض مقدار آن ستون را در رکورد‌هایی که از قبل موجود هستند False قرار دهد.

* قطعه کد با نمونه دکوراتور

|  |
| --- |
| @login\_required def do\_logout(request):  logout(request)  return HttpResponseRedirect('/') |

@login\_required یک تزئین‌کننده (decorator) است که منطق پایش ورود به سیستم را به view زیر خود اضافه می‌کند. در حقیقت کنترل می‌کند کسی که این تابع را فراخوانی می‌کند حتما وارد سیستم شده باشد.

* قطعه کد تعریف یک مدل

|  |
| --- |
| from django.db import models from django.contrib import auth from django.contrib.auth.models import User from django\_jalali.db import models as jmodels import jdatetime # CHOICES DEGREE\_CHOICES=[(1,u'کارشناسی ارشد') , (2,u'دکتری')] GENDER\_CHOICES=[(0,u'سرکار خانم'),(1,u'جناب آقای')] class Student(models.Model):  first\_name=models.CharField(verbose\_name='نام',max\_length=30)  last\_name=models.CharField(verbose\_name='نام خانوادگی',max\_length=30)  student\_number=models.CharField(max\_length=12,verbose\_name='شماره دانشجویی')  major=models.ForeignKey('Major',verbose\_name='رشته‌ی تحصیلی')  degree=models.IntegerField(choices=DEGREE\_CHOICES,verbose\_name='مقطع تحصیلی')  father\_name=models.CharField(verbose\_name='نام پدر',max\_length=30)  gender=models.IntegerField(choices=GENDER\_CHOICES,verbose\_name='جنسیت')  national\_id=models.CharField(verbose\_name='شماره ملی',max\_length=10)  has\_active\_request=models.BooleanField(default=False,verbose\_name='درخواست فعال دارد')  #...  user\_account=models.OneToOneField(User,verbose\_name='حساب کاربری', related\_name='student')  class Meta:  verbose\_name=u'دانشجو'  verbose\_name\_plural=u'دانشجویان'   def get\_info\_from\_user(self):  return u"{} {}".format(self.user\_account.first\_name,self.user\_account.last\_name)  @property  def info\_from\_self(self):  return u"{} {}".format(self.first\_name,self.last\_name)  @property  def full\_name(self):  return u"{} {} {}".format(self.get\_gender\_display(),self.first\_name,self.last\_name)  def \_\_str\_\_(self):  return u"{} دانشجوی {} رشته‌ی {}".format(self.full\_name,self.get\_degree\_display(),self.major) |

این کد مربوط به تعریف موجودیت دانشجو (student) است. در حقیقت یک کلاس پایتون به نام student داریم که از کلاس models.Model ارث‌بری دارد. کلاس Model شامل تمامی منطق‌هایی است که ORM نیاز دارد تا با این مدل کار کند. همچنین ماژول models شامل تعداد کلاس است که هر کدام از آن‌ها متناظر با یک جنس داده در ستون‌های جداول پایگاه‌داده هستند. به طور مثال first\_name که نمونه‌ای کلاس CharField است یک ستون از جنس کاراکتر خواهد بود که با توجه به پارامتر max\_length که به آن پاس داده شده است، سایز این ستون در پایگاه داده نیز ۳۰ کاراکتر خواهد بود. ForeignKey نیز مشخص می‌کند که این موجودیت به موجودیت دیگری که نامش را پاس می‌دهیم کلید خارجی (ارتباط یک به چند) دارد.

در ادامه با یک کلاس به نام Meta رو به رو می‌شویم که در درون کلاس Student تعریف شده است. این کلاس دربرگیرنده‌ی اطلاعات اضافی است که در سطح اپلیکیشن به آن‌ها نیاز داریم.

همچنین تعدادی تابع تعریف کرده ایم که کار با این مدل و یک سری منطق ابتدایی را برایمان راحت‌تر کند. به طور مثال تابع \_\_str\_\_ که یک تابع استاندارد در شی‌ءگرایی پایتون است برای تعیین نمود متنی شیء استفاده می‌شود. در حقیقت تعیین می‌کنیم اگر یک نمونه از کلاس Student را print کردیم به جای آن که مشخصات شیء را نمایش دهد یک مقدار به شکل رشته نمایش دهد مثل: علی رضایی دانشجوی کارشناسی ارشد رشته‌ی مهندسی عمران

دکوراتور @property باعث می‌شود تا بتوان از تابع به صورت یک فیلد عادی استفاده کرد.

نمونه کد‌های سمت کلاینت

* یک نمونه کد قالب پایه

|  |
| --- |
| {% load staticfiles %}  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge" />  <meta name="viewport" content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=7">  <meta name="language" content="en-EN" />  <meta name="author" content="Ashkan Taravati" />  <title>{% block title %} {% endblock%}</title>  <link rel="stylesheet" type="text/css" href="{% static 'common/styles/main.css' %}">  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">  <link href="https://cdn.rawgit.com/rastikerdar/vazir-font/v18.0.1/dist/font-face.css" rel="stylesheet" type="text/css" />  *<!--bootstrap-->*  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.css" integrity="sha384-Smlep5jCw/wG7hdkwQ/Z5nLIefveQRIY9nfy6xoR1uRYBtpZgI6339F5dgvm/e9B" crossorigin="anonymous">  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/js/bootstrap.min.js" integrity="sha384-o+RDsa0aLu++PJvFqy8fFScvbHFLtbvScb8AjopnFD+iEQ7wo/CG0xlczd+2O/em" crossorigin="anonymous"></script>  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/js/bootstrap.min.js"></script>  <script src="//code.jquery.com/jquery-1.11.1.min.js"></script>  <link rel="stylesheet" type="text/css" href="{% static 'common/js/main.css' %}"> {% block stylesheet %}{% endblock %}  </head>  <body>  <header>  <div class="row mrl-0">  <div class="col-md-12">    <h4><a class="nav-link">  سامانه رزرو اتاق دفاع  <img src="{% static 'common/img/Azad\_University\_logo.png' %}" style="height: 40px; width: 40px;">  </a></h4>  </div>  </div>  </header>  *<!-- HEADER END-->*  <nav class="navbar navbar-expand-lg navbar-light" style="background-color:#5ca9ff !important;">  <button class="navbar-toggle rtlDirection text-right d-xl-none d-lg-none" data-toggle="collapse" data-target=".navHeaderCollapse">  <i class="fa fa-bars fa-2x" style="color: aliceblue;"></i>  <span class="icon-bar"> </span>  <span class="icon-bar"> </span>  <span class="icon-bar"> </span>  <span class="icon-bar"> </span>  </button>  <div class="collapse navbar-collapse rightToleft navHeaderCollapse" id="navbarSupportedContent" style="direction: rtl !important; ">  <ul class="nav navbar-nav navbar-right text-right">  <li class="nav-item ">  </li>  <li class="nav-item ">  <a class="nav-link" href="{% url 'home' %}">  <i class="fa fa-home"></i> صفحه اصلی  </a>  </li>  <li class="nav-item">  <a class="nav-link" href="{% url 'dashboard' %}">  <i class="fa fa-graduation-cap"></i> سامانه دانشجو</a>  </li>  <li class="nav-item">  <a class="nav-link" href="/admin">  <i class="fa fa-user"></i>  ورود پرسنل </a>  </li>  <li class="nav-item">  <a class="nav-link" href="">  <i class="fa fa-question"></i> راهنما </a>  </li>  </ul>  </div>  </nav>  *<!-- LOGO HEADER END-->*  <div class="content-wrapper">  <div class="container col-10" >  <div class="title">  {% block heading %}{% endblock heading %}  </div>  <div >  {% block content %}{% endblock %}  </div>  </div>  </div>  *<!-- CONTENT-WRAPPER SECTION END-->*  <footer>  <div >  <div class="row mrl-0">  <div class="col-md-12">  دانشگاه آزاد اسلامی واحد تهران مرکزی | توسعه یافته توسط گروه مهندسی کامپیوتر  </div>  </div>  </div>  </footer>  *<!-- FOOTER SECTION END-->*  *<!-- JAVASCRIPT AT THE BOTTOM TO REDUCE THE LOADING TIME -->*  *<!-- CORE JQUERY SCRIPTS -->*  *{% comment %} <script src="assets/js/jquery-1.11.1.js"></script> {% endcomment %}*  *<!-- BOOTSTRAP SCRIPTS -->*  *{% comment %} <script src="assets/js/bootstrap.js"></script> {% endcomment %}*  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.js"></script>  <script src="{% static 'student/main.js' %}"></script>  {% block javascript %}{% endblock %}  </body>  </html> |

این کد قالب اصلی است که سایر قالب‌ها میتوانند در درون آن قرار گیرند. بیشتر بخش‌های این قالب مانند یک فایل html عادی است اما در بعضی قسمت‌ها از دستورات DTL استفاده شده است.

* کد مربوط به درخواست AJAX

|  |
| --- |
| function submitIdAJAX(sender, endpoint) {  let did = $(sender).data('id');  $.ajax({  url: endpoint,  data: {  id: did  },  dataType: 'JSON',  method: 'POST',  success: function (result) {  console.log('success:' + result.msg);  ;debugger   let noticeBox = $('#notice');  noticeBox.removeClass('alert-danger');  noticeBox.addClass('alert-info');  noticeBox.removeClass('d-none');  noticeBox.html(result.msg);  },  error: function () {  console.log('ERROR:submitReservation failed');  let noticeBox = $('#notice');  noticeBox.removeClass('alert-info');  noticeBox.addClass('alert-danger');  noticeBox.removeClass('d-none');  noticeBox.html('<strong>خطا</strong>');  }  }) } |

این کد جاوااسکریپت به این شکل کار می‌کند که مقدار صفت data-id دکمه‌ی فراخواننده‌ی خود را می‌گیرد و به آدرس مشخص شده (که بیانگر view مورد نظر است) با استفاده از توابع موجود در کتابخانه‌ی jQuery یک درخواست POST ارسال می‌کند و با توجه به نتیجه‌ی درخواست که به شکل یک شیء JSON باز می‌گردد، اگر موفقیت‌آمیز باشد باکس پیغام را به رنگ آبی و پیام موفقیت را منعکس کرده و در صورت شکست، باکس را قرمز کرده و پیام خطا را نمایش می‌دهد.

* کد نمونه قالب

|  |
| --- |
| {% extends "public/public\_layout.html" %}  {% block title %}  تیتر صفحه  {% endblock title %}  محتوای آزمایشی  {% block content %}  {% endblock %} |

این کد یک مثال ساده و آزمایشی است که در صورت پردازش به محتویات فایل public\_layout.html اضافه می‌شود.

* کد نمونه دستورات DTL شبه برنامه‌نویسی در قالب

|  |
| --- |
| <table class="table table-hover " >  <thead class=" row-blue-bg">  <tr class="filters">  <th >مشخصه</th>  <th >زمان</th>  <th > مکان</th>  <th > موضوع</th>  <th >رشته</th>  <th >توضیحات</th>  <th></th>  </tr>  </thead>  <tbody>  {% for schedule in schedules %}  {% if schedule.designated\_defense\_time %}      <tr>  *{% comment %} <td>{{forloop.counter}}</td> {% endcomment %}*  <td>{{schedule.id}}</td>  <td>{{schedule.designated\_defense\_time}}</td>  <td>{{schedule.designated\_defense\_time.defense\_place}}</td>  <td>{{schedule.subject}}</td>  <td>{{schedule.major}}</td>  <td>{{schedule.info}}</td>  <td><a href="{% url 'defense\_announcement' dsid=schedule.id %}">مشاهده جزئیات</a></td>    </tr>  {% endif %}  {% endfor %}  </tbody>  </table> |

در این دستورات یک جدول به صورت پویا ایجاد کرده ایم، به این صورت که به ازای هر یک از عناصر واکشی شده از جستاری که از view به سمت قالب پاس داده شده، مقادیر موجود در یک آبجکت schedule را که یک رکورد از جدول «جلسات دفاع» است در مکان مناسب خود در میان تگ‌های html قرار داده‌ایم تا به مناسب‌ترین شکل نمایش داده شود. تقریبا مشابه آن است که در یک حلقه‌ی for یکایک عناصر یک آرایه‌ را چاپ کنیم.

روش راه‌اندازی سامانه در محیط توسعه

به منظور راه‌اندازی پروژه در محیط توسعه می‌بایست ابتدا محیط مجازی را فعال کنیم. برای این منظور ابتدا دستور زیر را (با فرض این که در پوشه‌ی والد باشیم) در ترمینال وارد می‌کنیم:

|  |
| --- |
| source venv/bin/activate |

این دستور با فعال کردن محیط مجازی، یک سری از متغیر‌های محیطی را بازتعریف می‌کند. مثلا باعث می‌شود سیستم از مفسری که در سیستم عامل موجود است و انباره‌های اصلی سیستم استفاده نکند و از موارد موجود در پوشه‌ی محیط مجازی استفاده کند.

در صورتی که اولین بار است سامانه را راه‌اندازی می‌کنیم دو کار لازم است انجام شود.

1. نصب پکیج‌های لازم برای اجرای سامانه

برای این منظور دستورات زیر را در ترمینال وارد می‌کنیم

|  |
| --- |
| cd TDRSS pip install -r requirements.txt |

دستور اول دایرکتوری کنونی را به پوشه‌ی پروژه تغییر میدهد و دستور دوم فایل requirements.txt را که شامل نام و نسخه‌های مورد نظر پکیج‌های مورد نیاز است به عنوان ورودی به دستور نصب ابزار مدیریت پکیج پایتون می‌دهد تا یکی یکی آن‌ها را دانلود و نصب کند.

1. مهاجرت و ساخت پایگاه داده

در این مرحله با وارد کردن دستور زیر، مدل‌ها به شکل جداول در یک پایگاه داده‌ی sqlite ایجاد می‌شوند.

|  |
| --- |
| python manage.py makemigrations student python manage.py migrate |

دستور اول یک سری فایل migration ایجاد می‌کند و دستور دوم یکایک آن‌ فایل‌ها رو به ترتیب اجرا و اعمال می‌کند.

حال می‌توانیم سامانه را اجرا کنیم. پس دستور زیر را وارد می‌کنیم:

|  |
| --- |
| python manage.py runserver |

این دستور سرور توسعه (Django Development Server) را فعال می‌کند. پیغام موفقیت آن به شکل زیر خواهد بود:

|  |
| --- |
| Performing system checks...  System check identified no issues (0 silenced). August 22, 2018 - 15:38:40 Django version 1.11, using settings 'TDRRS.settings' Starting development server at http://127.0.0.1:8000/ Quit the server with CONTROL-C. |

با توجه به این پیام، حال سامانه در آدرس محلی <http://127.0.0.1:8000/> قابل دسترسی است. البته می‌توان صراحتا پورت دیگری را مشخص کرد اما پورت پیش‌فرض سرور توسعه‌ی جنگو ۸۰۰۰ می‌باشد. توجه داریم که این سرور برای محیط عملیاتی مناسب نیست.

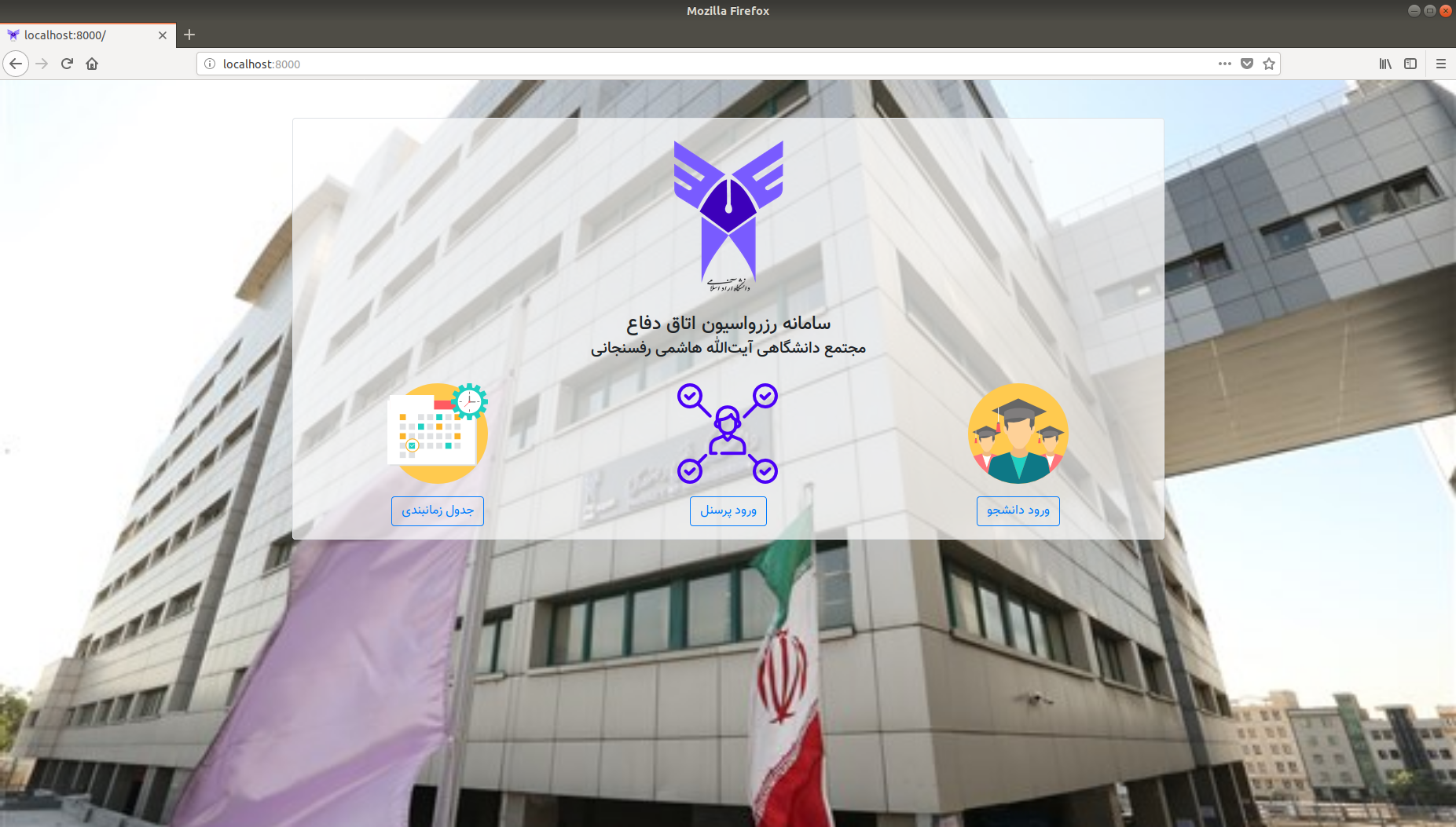
اگرچه سیستم کار می‌کند اما برای ورود به پنل مدیریت لازم است یک ابرکاربر در پایگاه داده وارد کنیم. برای این منظور بار دیگر از manage.py استفاده می‌کنیم. ابتدا Ctrl+C را فشار داده و دستور زیر را وارد می‌کنیم:

|  |
| --- |
| python manage.py createsuperuser Username: ali Email address: ali@domain.ir Password: |

جنگو با پرسیدن اطلاعاتی کاربری با دسترسی کامل برایمان ایجاد می‌کند که بعدا با مراجعه به پنل ادمین می‌توان وارد مدیریت سامانه شد.

راهنمای کاربری سامانه

پس از ورود به سامانه با این صفحه که صفحه‌ی اصلی آن می‌باشد مواجه می‌شویم:



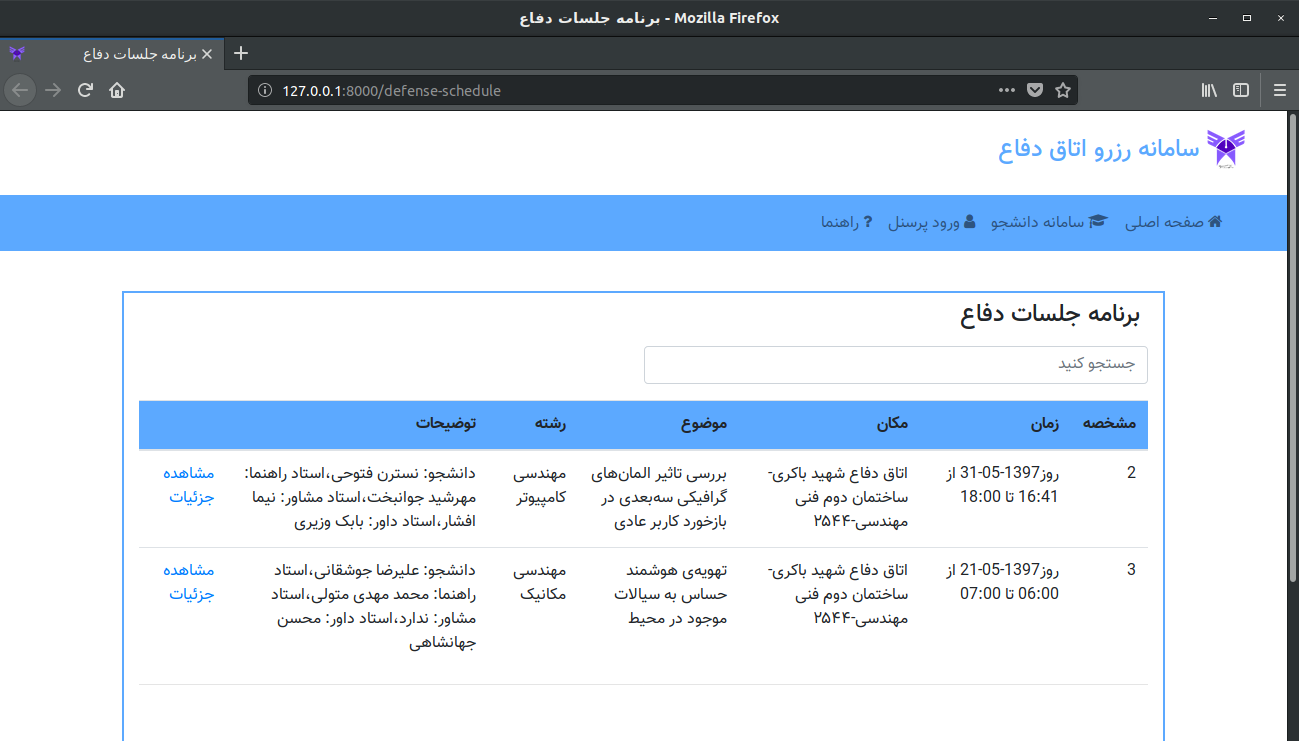
تصویر 11 صفحه‌ی اصلی سامانه

راهنمای بخش جدول زمان‌بندی

این بخش قابل استفاده‌ی عموم کاربران می‌باشد و محدودیتی برای ورود به آن وجود ندارد.

برای ورود به این بخش می‌بایست بر روی دکمه‌ی «جدول زمان‌بندی» کلیک نماییم.

با ورود به قسمت «جدول زمان‌بندی» با صفحه‌ی زیر رو به رو می‌شویم:



تصویر 12 جدول زمانبندی جلسات

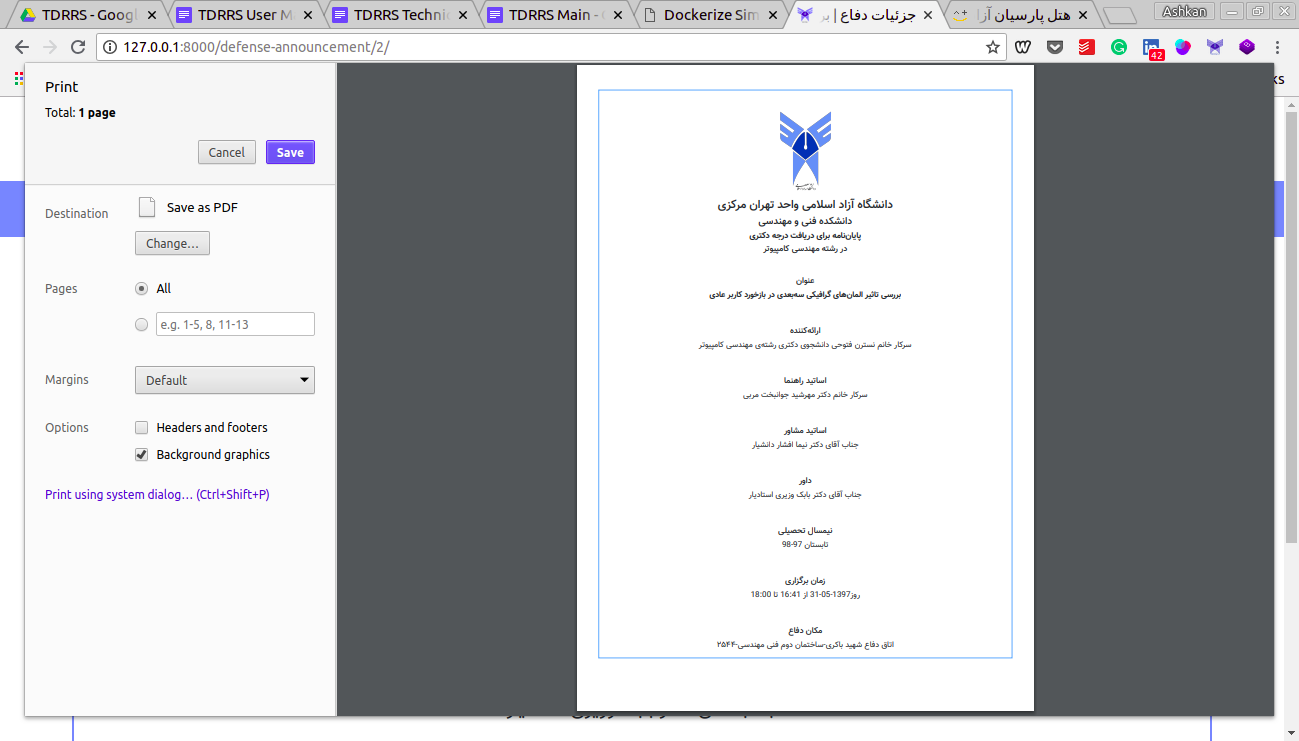
در این صفحه‌ فهرست تمام جلسات دفاعیه‌ی آتی که زمان آن‌ها قطعی شده است قابل مشاهده است.

برای مشاهده‌ی جزئیات دفاعیه‌ی مورد نظر میتوانید با کلیک بر روی پیوند «مشاهده‌ی جزئیات» وارد صفحه‌ی جزئیات آن شوید.



تصویر 13 اعلامیه‌ی دفاع

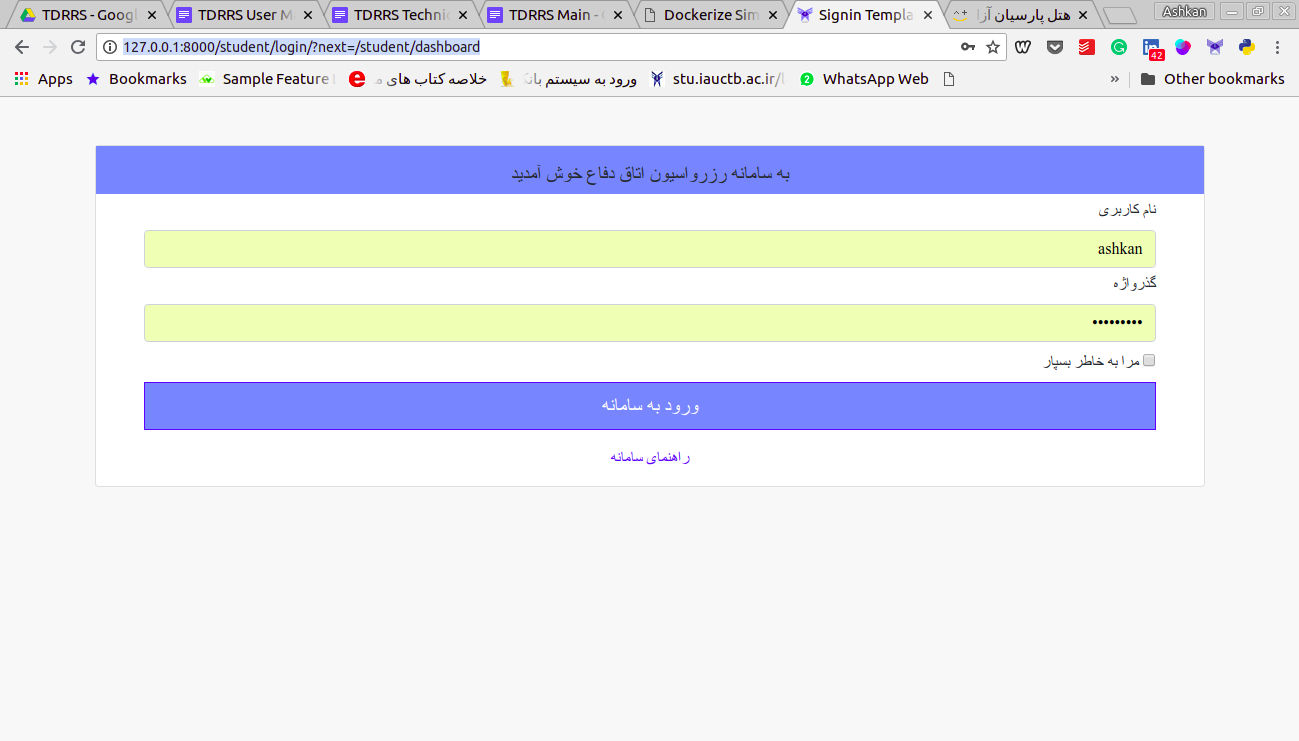
این صفحه دارای قابلیت چاپ می‌باشد به این معنا که با دادن دستور چاپ به مرورگر (Ctrl+P) یا فشردن دکمه‌ی «چاپ» در نوار ابزار بالا، صفحه‌ تبدیل به یک اعلامیه‌ی استاندارد آماده‌ی چاپ می‌شود.



تصویر 14 چاپ اعلامیه

راهنمای بخش دانشجو

با کلیک بر روی دکمه‌ی «ورود دانشجو» از ما خواسته می‌شود تا وارد سیستم شویم.



تصویر 15 صفحه‌ی ورود

با وارد کردن اطلاعات کاربری صحیح به صفحه‌ی داشبورد سامانه‌ی دانشجو هدایت می‌شویم که به شکل زیر است



تصویر 16 داشبورد دانشجو



تصویر 17 داشبورد دانشجو - پیگیری درخواست

این صفحه از سه قسمت تشکیل شده است:

1. مشاهده‌ی اطلاعات هویتی دانشجو
2. مشاهده‌ی مشخصات پروپوزال و دفاعیه‌ی جاری دانشجو
3. فهرست درخواست‌‌هایی که دانشجو تا کنون به منظور هماهنگی زمان و مکان تشکیل جلسه دفاع خود ثبت کرده است

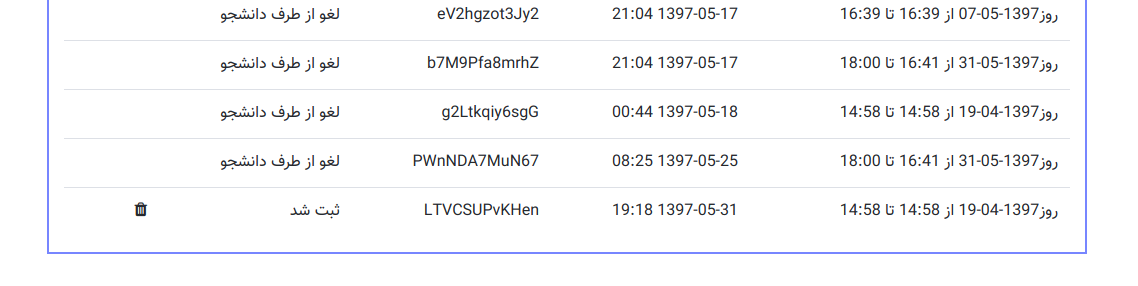
در بخش درخواست‌ها می‌توان با کلیک روی دکمه سطل آشغال درخواست مربوطه را لغو کرد. البته این در شرایطی است که درخواست پیش‌تر لغو نشده باشد

یکی از بخش‌های دیگر سامانه دانشجو صفحه‌ی «زمان‌های قابل رزرو» می‌باشد در این صفحه فهرستی از زمان-مکان ها (یعنی چه زمانی، در چه تاریخی، در کدام اتاق دفاع) که دانشجوی وارد شده با توجه به رشته‌ی و سایر قیود در ترم جاری می‌تواند رزرو کند قابل مشاهده است. دانشجو می‌تواند با فشردن دکمه‌ی ثبت در مقابل آیتم مورد نظر آن را به رزرو خود در بیاورد.

در صورت موفقیت پیام موفقیت در صفحه مشاهده خواهد شد و در صورت صرف نظر همانطور که در بالا گفته شد می‌تواند عمل نماید.



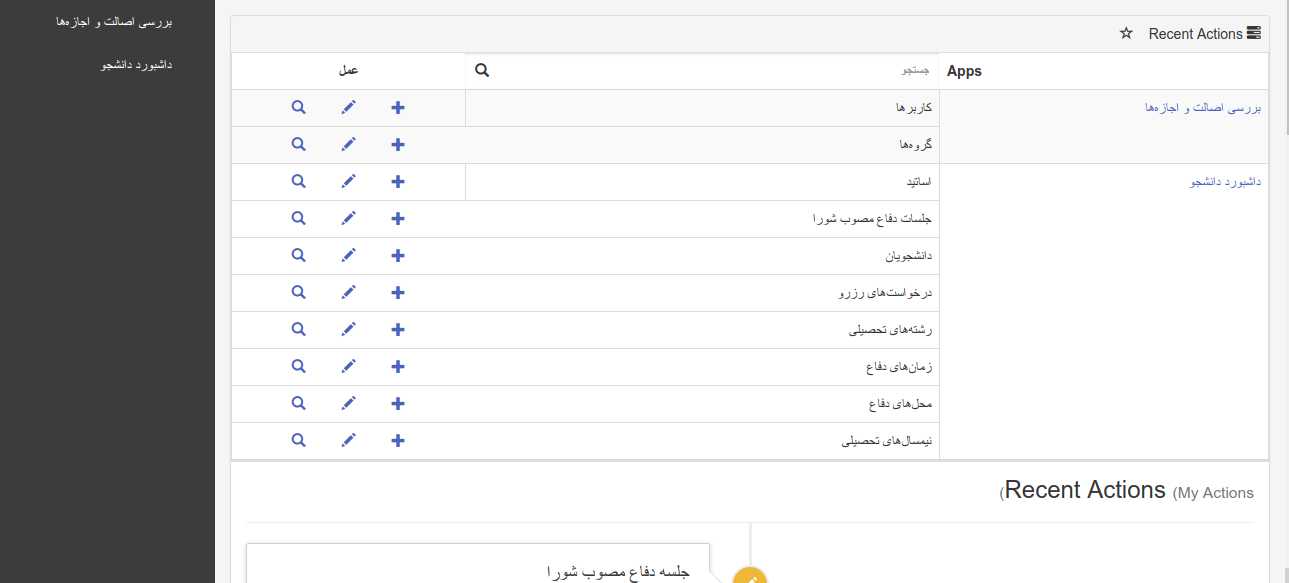
تصویر 18صفحه‌ی زمان‌ها قابل رزرو



تصویر 19 لغو رزرو

راهنمای بخش مدیریت

این بخش امکانات زیادی از قبیل مدیریت کاربران، مدیریت دسترسی‌ها، انجام عملیات CRUD بر روی جداول پایگاه‌داده‌ها و پایش آخرین رویداد‌های سامانه را دارا می‌باشد که مجوز استفاده از همه‌ی این امکانات برای کاربر از نوع کارمند پژوهش توجیه نداشته و مناسب راهبران سامانه در اداره‌ی فناوری اطلاعات می‌باشد. فلذا می‌بایست دسترسی‌های ایشان و حتی سایر نقش‌های کاربری در صورت نیاز به درستی تعیین گردد.



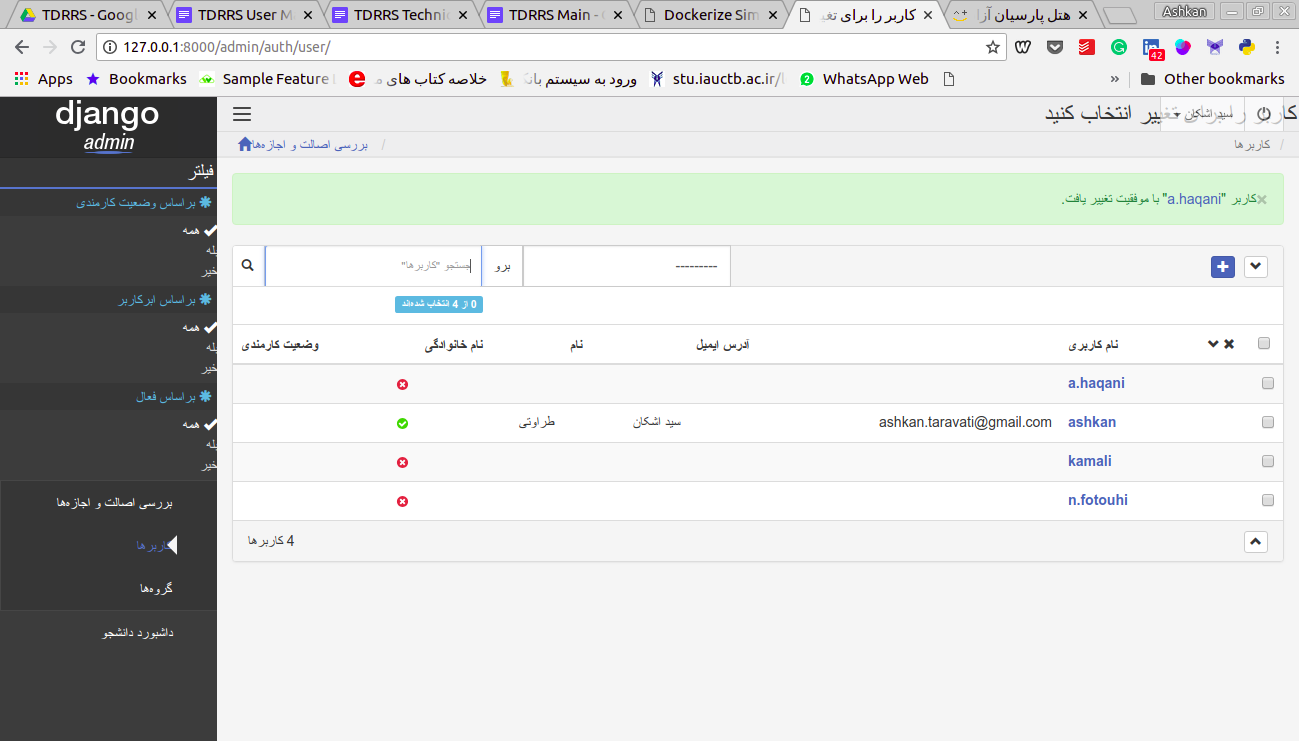
تصویر 20 داشبورد مدیریت



تصویر 21 رویداد‌های اخیر

بخش بررسی اصالت و اجازه‌ها شامل دو بخش کاربرها و گروه‌ها می‌باشد

در بخش کاربرها می‌توان به مدیریت کاربران سامانه و ویرایش دسترسی‌ها و سایر اطلاعات آن‌‌ها پرداخت



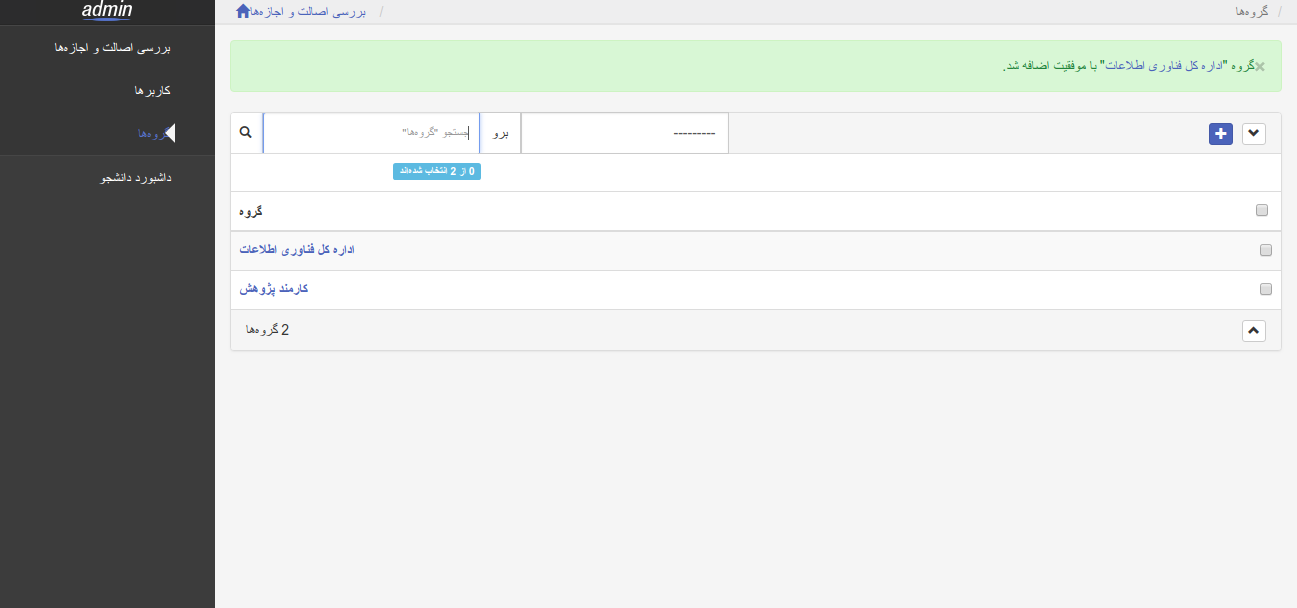
تصویر 22 مدیریت کاربران

به طور مثال صفحه‌ی ویرایش یکی از کاربران به این شکل است:



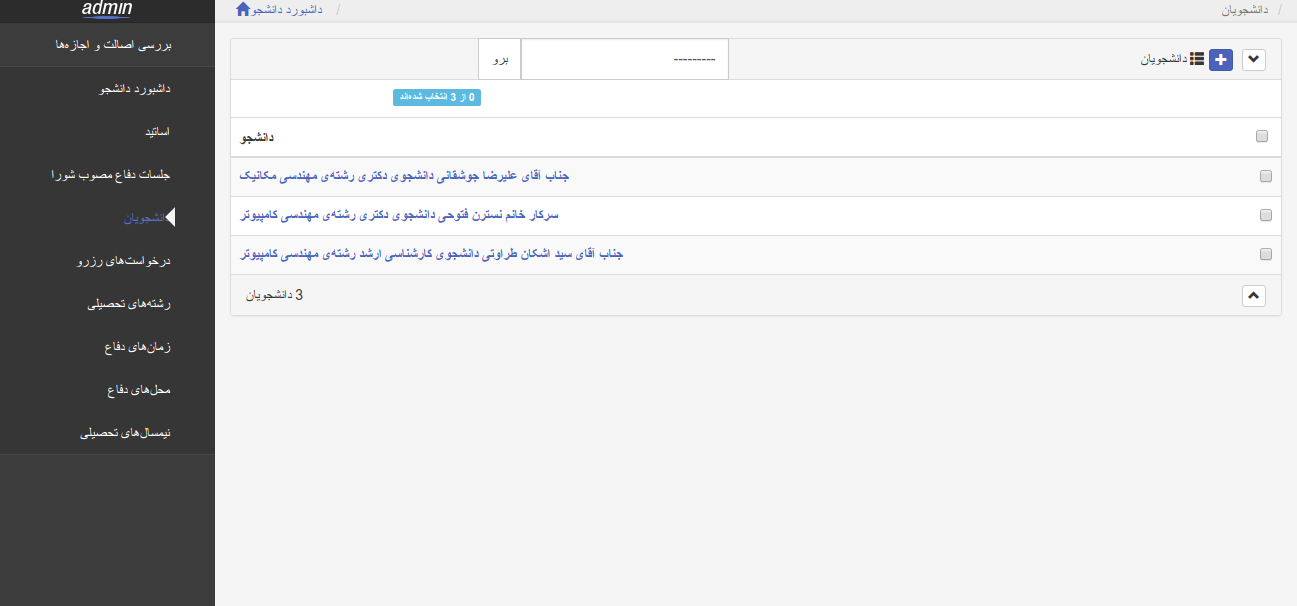
تصویر 23 ویرایش کاربر

در بخش گروه‌ها می‌توان به ازای نقش‌های مختلف در سامانه گروه‌هایی تعریف نمود. هر گروه دارای مجموعه‌ای از دسترسی‌ها (اجازه‌ها) می‌باشد که به راحتی می‌توان با افزودن یک کاربر به یک گروه دسترسی‌های مربوط به آن گروه را به کاربر نسبت داد.



تصویر 24 گروه‌ها (نقش‌ها)

در دسته‌ی آیتم‌های داشبورد‌ دانشجو فهرستی از جداول مربوط به سامانه‌ی دانشجو قابل رؤیت است و کاربر می‌تواند بر حسب دسترسی‌های خود اقدام به ویرایش، ثبت، مشاهده و حذف رکورد‌هایی از جداول نماید. نمونه‌هایی از صفحات این بخش در ادامه مشاهده می‌شود:



تصویر 25 مدیریت دانشجویان

منابع و مراجع

* مهندسی نرم‌افزار، راجر اس پرسمن، ترجمه دکتر محمدمهدی سالخورده حقیقی، انتشارات خراسان، ویرایش هفتم، ۷۸۴ صفحه
* کتاب مفاهیم بنیادی پایگاه داده ها، محمد تقی روحانی رانکوهی، انتشارات جلوه، ویراست چهارم، ۸۰۸ صفحه
* UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition, Martin Fowler, Addison-Wesley, 2004, 175 pages
* The New Relational Database Dictionary: Terms, Concepts, and Examples, By C.J. Date, “O'Reilly Media, Inc.", 2015, ISBN 1491951710, 9781491951712, 450 pages
* Two Scoops of Django 1.11: Best Practices for the Django Web Framework, Daniel Roy Greenfield, 4th edition, Two Scoops Press, 2017, ISBN: 0692915729, 9780692915721,519 pages
* Computer Networking: A Top-Down Approach, James F. Kurose, Keith W. Ross,5th edition, Pearson Education, 2010, ISBN 0131365487, 9780131365483, 888 pages

وبسایت‌های مفید:

* https://docs.djangoproject.com/en/1.11/
* https://readthedocs.org/projects/django/
* <https://www.mspsoft.com>
* <https://www.djangoproject.com/>
* <https://www.w3schools.com/bootstrap4/>
* <https://whatis.techtarget.com>
* <https://www.javascript.com/>
* <https://www.python.org/>
* <https://getbootstrap.com/>
* <https://searchnetworking.techtarget.com>
* <https://www.tutorialspoint.com>
* http://[www.uml.org/](http://www.uml.org/)

