طراحی نرم افزار با زبان مدلسازی UML

Weblog-team

APRIL 22, 2025

MOHAMMADHASAN-MEHRABI
MOHAMMAD-AHMADZAEH
دانشگاه ملی ومهارت ایت الله خامنه ای
Github:@mohammadhasanmehrabi

۱ .معرفی پروژه

در این پروژه ما قصد داریم که یک وبلاگ برای معرفی تیم برنامه نویسی نویستا بسازیم که علاوه بر اینکه کاربران میتوانند در اعضای تیم رو شناسایی کنند پروژه هایی که این ساخته و مقالات این تیم رو هم برسی کنند و برای همکاری میتوانند ثبت نام کنند و با تیم نویستا ارتباط بگیرند

۱/۱ هدف از مدلسازی باUML

- زبان UML 2.0 استاندارد صنعتي مستندسازی و تحلیل سیستم‌های نرم‌افزاری است که هم مدل‌های ساختاری (ساختار کلاس‌ها، مؤلفه‌ها، استقرار) و هم مدل‌های رفتاری (جریان‌ها، تعاملات، حالت‌ها) را پوشش می‌دهد با استفاده از UML می‌توانیم پیش از نوشتن حتی یک خط کد، معماری دقیق سیستم، اجزای کلیدی و تعاملات آن‌ها را به‌صورت رسمی و قابل‌مذاکره با ذی‌نفعان (مشتریان، تیم توسعه) مستند کنیم.

۱/۲ عملکردهای اصلی نرم‌افزار

1. **ثبت‌نام (Sign-up) و ورود :(Login)** احراز هویت کاربران برای دسترسی به بخش‌های محافظت‌شده
2. **بخش «درباره ما :»**نمایش اطلاعات تیم، مأموریت و تاریخچه.
3. **بخش «ارتباط با ما :»**فرم ارسال پیام و اطلاعات تماس.
4. **بخش «کارهای ما :»**نمایش نمونه‌کارها (Portfolio) به‌همراه توضیحات پروژه‌ها.
5. **بخش «مقالات ما :»**فهرست و جزئیات پست‌های وبلاگی با امکان دسته‌بندی و جستجو

۲ .معرفی زبانUML

UML (Unified Modeling Language)زبانی گرافیکی-متنی است که مفاهیم مهندسی نرم‌افزار را در قالب انواع نمودارها نمایش می‌دهد. این زبان به‌منظور یکپارچه‌سازی بهترین شیوه‌های پیشین مدلسازی شیءگرا(مانندOMT ، Boochو OOSE) طراحی شده و اکنون طیف وسیعی از جنبه‌های طراحی، تحلیل و مستندسازی سیستم را در بر می‌گیرد.

۲/۱ اهداف و قابلیت‌ها

- **ارتباط میان ذی‌نفعان** UML **:**زبانی مشترک است تا تحلیلگران، توسعه‌دهندگان، مدیران پروژه و مشتریان بتوانند روی مدل‌ها توافق کنند.
- **اعتبارسنجی معماری :**به کمک نمودارها می‌توان پیش از کدنویسی معماری سیستم را بررسی و خطاها را شناسایی کرد.
- **مستندسازی رسمی :**خروجی‌های UML به‌عنوان مستندات رسمی پروژه قابل نگهداری و پیگیری در طول چرخه حیات نرم‌افزار هستند.

۲/۲ جنبه‌های تحت پوششUML 2.0

در UML 2.0 نمودارها به سه دسته اصلی تقسیم می‌شوند:

- **مدل‌های ساختاری(Structure Diagrams)**
  1. Class Diagram
  2. Component Diagram
  3. Deployment Diagram

4. Object Diagram
5. Package Diagram
6. Composite Structure Diagram
7. Profile Diagram

- **مدل‌های رفتاری(Behavior Diagrams)**

1. Use Case Diagram
2. Activity Diagram
3. State Machine Diagram
4. Sequence Diagram
5. Communication Diagram
6. Interaction Overview Diagram
7. Timing Diagram

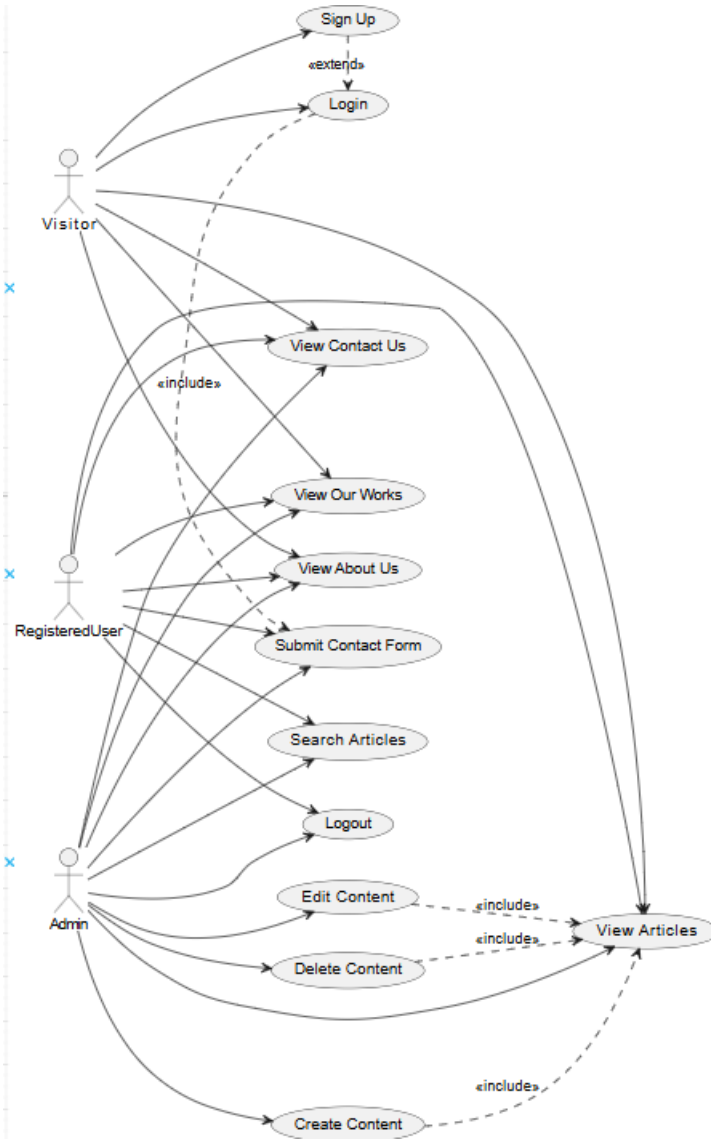
۳ .جمع‌آوری نیازمندی‌ها

۳/۱ شناسایی بازیگران(Actors)

بازیگر	توضیح
Visitor	کاربری که بدون ورود می‌تواند بخش‌های عمومی (درباره ما، مقالات، نمونه‌کارها) را ببیند.
RegisteredUser	کاربری که پس از ثبت‌نام/ورود می‌تواند به امکانات اضافه (نظردهی، ارسال فرم ارتباط) دسترسی داشته باشد.
Admin	مدیر سیستم که می‌تواند محتوا اضافه/ویرایش/حذف کند) خارج از محدوده اینUse Case ها.(

۳/۲ شناساییUse Case های اصلی

Use Case	Actor	هدف
Sign Up	Visitor	ثبت‌نام کاربر جدید در سیستم برای دسترسی محافظت‌شده .
Login	Visitor	احراز هویت کاربر برای ورود به حساب کاربری.
View About Us	Visitor	مشاهده اطلاعات تیم و مأموریت
View Contact Us	Visitor	مشاهده فرم و اطلاعات تماس
Submit Contact Form	RegisteredUser	ارسال پیام از طریق فرم «ارتباط با ما«
View Our Works	Visitor	مشاهده نمونه‌کارهای تیم
View Articles	Visitor	مرور فهرست و جزئیات مقالات
Search Articles	RegisteredUser	جستجوی مقاله بر اساس کلمه‌کلیدی



۳٫۴ توضیح جریان‌ها برایUse Case های کلیدی

Use Case: Sign Up

- **Actor:** Visitor
- **Precondition:**کاربر روی صفحه ثبت‌نام قرار دارد.
- **Postcondition:**حساب کاربری جدید ایجاد و کاربر به صفحه پروفایل هدایت می‌شود.
- **Main Flow:**
  1. کاربر فرم ثبت‌نام را پر می‌کند (ایمیل، گذرواژه).
  2. سیستم ورودی‌ها را اعتبارسنجی می‌کند.
  3. در صورت موفقیت، حساب جدید ساخته می‌شود و ایمیل تأیید ارسال می‌شود.
  4. کاربر وارد سیستم می‌شود و به داشبورد هدایت می‌شود.
- **Alternate Flows:**
  - 2a. اگر اعتبارسنجی شکست خورد (مثلاً فرمت ایمیل نامعتبر)، پیام خطا نمایش داده می‌شود و کاربر مجدداً فرم را ویرایش می‌کند.
  - 3a. اگر ایمیل قبلاً ثبت شده باشد، پیام «کاربر وجود دارد» نمایش داده می‌شود.

Use Case: Login

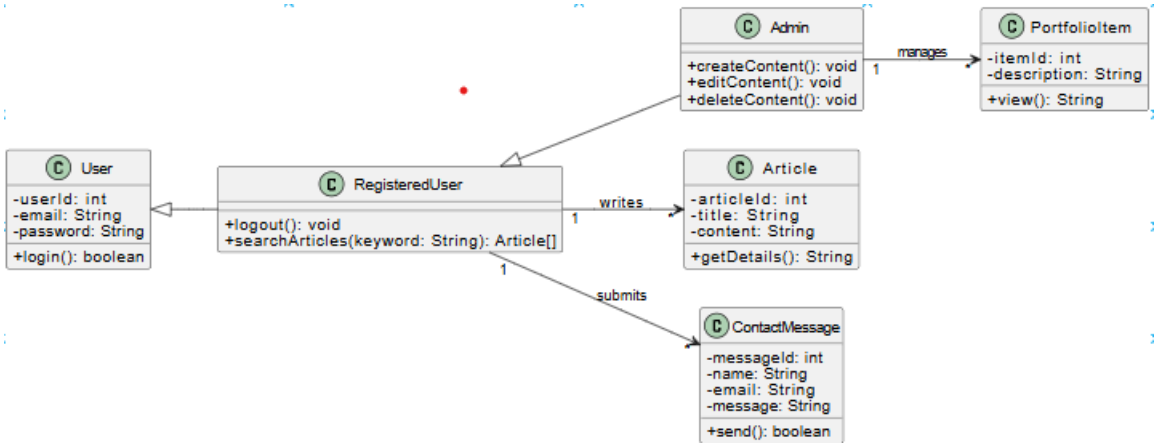
- **Actor:** Visitor
- **Precondition:**کاربر دارای حساب ثبت‌شده باشد.
- **Postcondition:**کاربر احراز هویت شده و به صفحه اصلی/داشبورد دسترسی پیدا می‌کند.
- **Main Flow:**
  1. کاربر نام‌کاربری و گذرواژه را وارد می‌کند.
  2. سیستم اعتبارسنجی می‌کند.
  3. در صورت موفقیت، جلسه (Session) ایجاد و کاربر وارد می‌شود.
- **Alternate Flows :**
  - 2a. در صورت اشتباه بودن اطلاعات، پیام خطا نمایش داده می‌شود و کاربر می‌تواند مجدداً تلاش کند.

Use Case: Submit Contact Form

- **Actor:** RegisteredUser
- **Precondition :** کاربر به‌حساب خود وارد شده باشد و صفحه «ارتباط با ما» را مشاهده کند.
- **Postcondition :** پیام کاربر در پایگاه‌داده ذخیره و ایمیل اطلاع‌رسانی به مدیر ارسال شود.
- **Main Flow:**
  1. کاربر فیلدهای فرم (نام، ایمیل، موضوع، پیام) را پر می‌کند.
  2. سیستم ورودی‌ها را اعتبارسنجی می‌کند (غیرخالی بودن فیلدها).
  3. در صورت صحت، پیام ذخیره و تأیید ارسال را نمایش می‌دهد.

- **Alternate Flows :**
  - 2a. در صورت خالی بودن فیلد ضروری، پیام خطا نمایش داده می‌شود.

۴/۱ نمودار کلاس (Class Diagram)



User: نماینده‌ی کاربران عام

RegisteredUser: ارث‌بری از User با دسترسی‌های اضافی

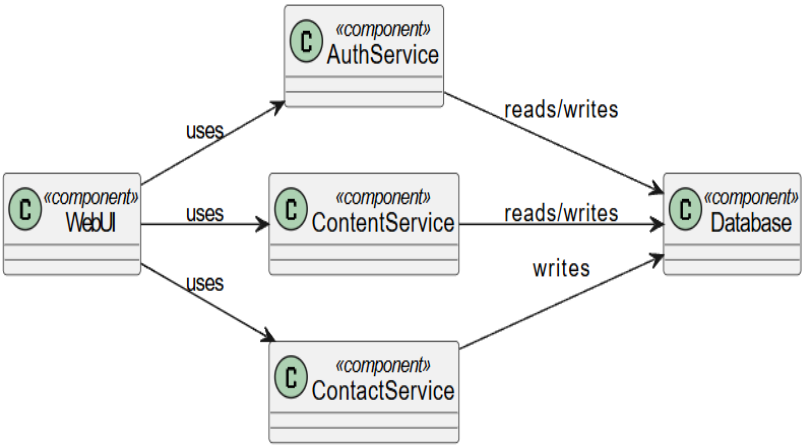
Admin: ارث‌بری از RegisteredUser با قابلیت‌های مدیریت محتوا

Article: نمایش پست‌های وبلاگ

PortfolioItem: نمونه‌کارهای تیم

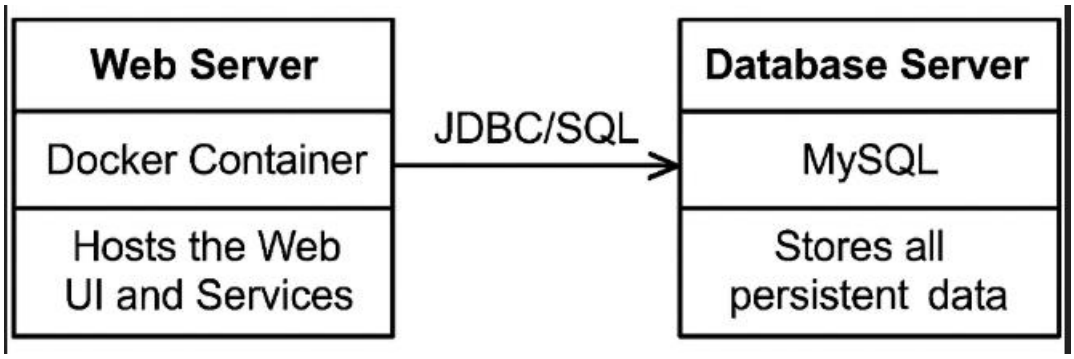
ContactMessage: پیام‌های ارسالی از فرم «ارتباط با ما»

۴/۲ نمودار مؤلفه (Component Diagram)

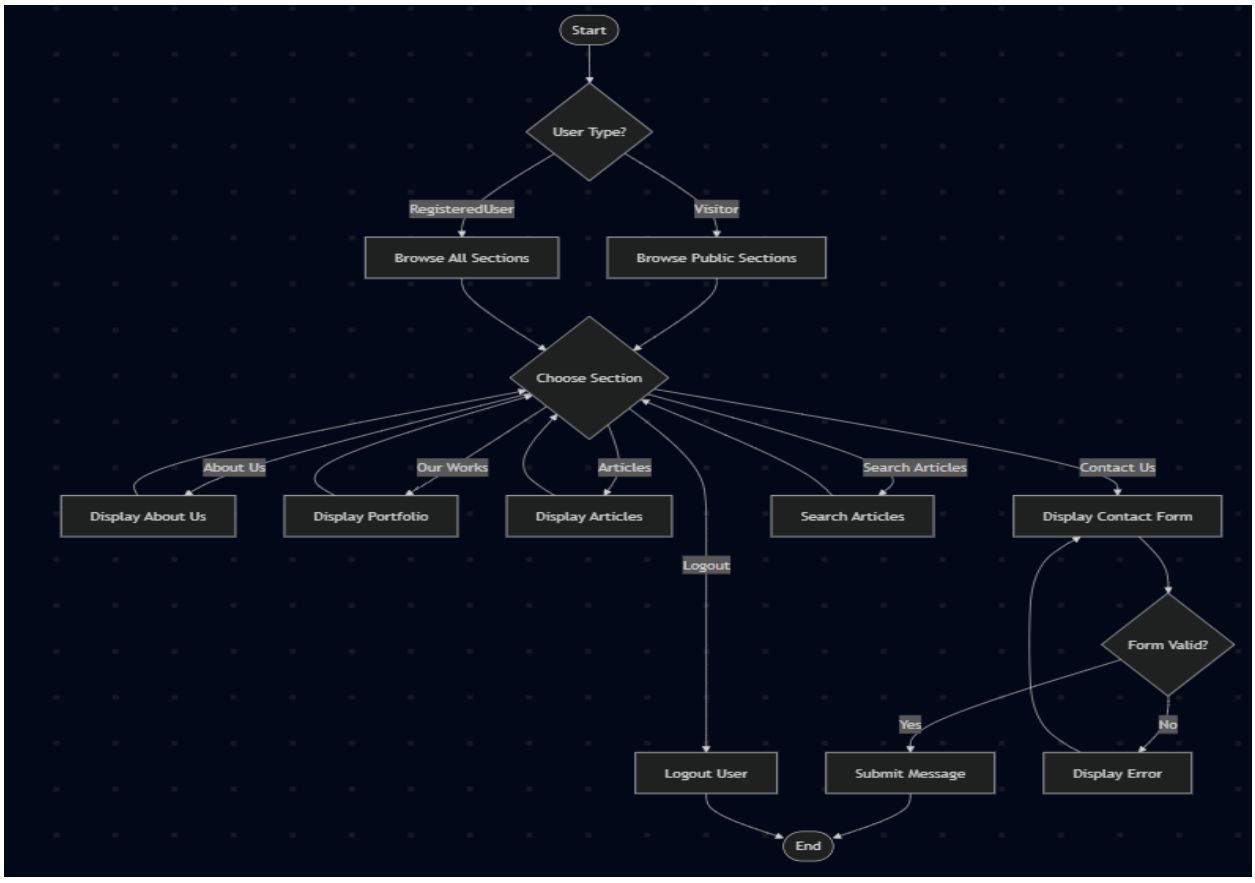


- **Web UI**: رابط کاربری وب
- **Auth Service**: مدیریت ورود/ثبت‌نام
- **Content Service**: مدیریت مقالات و نمونه‌کارها
- **Contact Service**: پردازش فرم «ارتباط با ما»
- **Database**: پایگاه‌داده‌های MySQL یا

۴/۳ نمودار استقرار(Deployment Diagram)



۵/۱ نمودار فعالیت(Activity Diagram)



۱ .تعیین نوع کاربر

در آغاز با گره تصمیم می‌فهمیم کاربر **Visitor**است یا **RegisteredUser**؛ این تصمیم مسیرهای بعدی را تعیین می‌کند .

۲ .مرور بخش‌ها

- **مرور بخش‌های عمومی :**وقتی کاربر Visitor است، فقط بخش‌های عمومی (About Us, Our Works, Articlesو Contact Us) قابل مشاهده‌اند .
- **مرور همه بخش‌ها** RegisteredUser **:**علاوه بر بخش‌های عمومی، به جستجوی مقالات و گزینهٔ Logout دسترسی دارد .

۳ .انتخاب بخش

گره انتخاب بخش تمام گزینه‌های ممکن را پیشنهاد می‌کند:

- About Us
- Our Works
- Articles
- Contact Us
- Search Articles
- Logout

این گره کاربر را به اکتیویتی مربوطه می‌فرستد .

۴ .نمایش محتوا

هر اکتیویتی نمایش (مثل «نمایش دربارهٔ ما»، «نمایش نمونه‌کارها» و «نمایش مقالات») صرفاً محتوای مربوطه را به کاربر می‌دهد و سپس بازمی‌گردد به «انتخاب بخش» تا ادامهٔ گردش امکان‌پذیر باشد .

۵ .جستجوی مقالات

RegisteredUser می‌تواند با وارد کردن کلیدواژه در اکتیویتی «جستجوی مقالات»، فهرستی از مقالات مرتبط را دریافت کند و پس از آن باز به «انتخاب بخش» بازگردد .

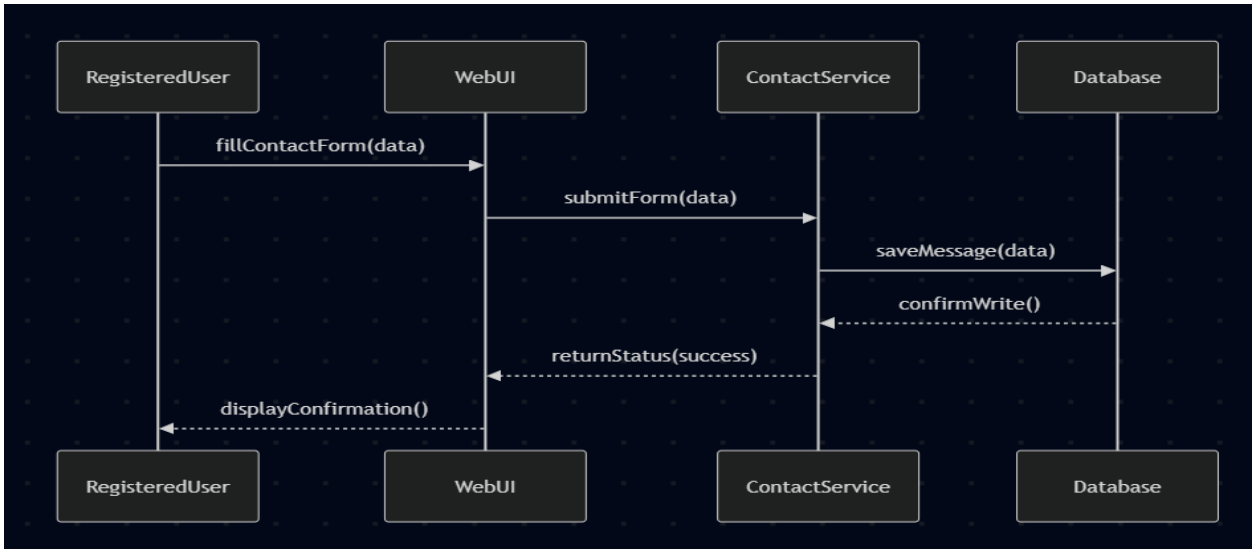
۶ .نمایش فرم تماس و اعتبارسنجی

- »نمایش فرم تماس» اکتیویتی نمایش فیلدهای فرم را بر عهده دارد.
- در «اعتبارسنجی فرم» بررسی می‌شود که همهٔ فیلدهای ضروری پر شده و قالب ورودی‌ها صحیح است (مانند ایمیل).
  - اگر **بله**، پیام ارسال شده و فرایند پایان می‌یابد.
  - اگر **خیر**، پیام خطا نمایش و فرم دوباره نمایش داده می‌شود تا کاربر اصلاح کند .

۷ .خروج(Logout)

انتخاب «خروج از سیستم» به گره پایان ((*)) می‌رود و فرایند کاملاً خاتمه می‌یابد

۵٫۲ نمودار توالی(Sequence Diagram)



تشریحParticipants

1. **RegisteredUser (RU)**
شرکت‌کننده‌ای است که از قبل در سیستم ثبت‌نام و احراز هویت شده است و نقش آغازگر (initiator) تعاملات فرم تماس را برعهده دارد.
2. **WebUI (UI)**
رابط کاربری وب که فرم تماس را به کاربر نمایش می‌دهد و داده‌های ورودی او را دریافت و به سرویس‌های پشت‌صحنه می‌فرستد
3. **ContactService (CS)**
مؤلفه‌ای در لایهٔ منطق تجاری (business logic) که مسئول اعتبارسنجی فرم و هماهنگی با لایهٔ دسترسی به داده (Database) برای ذخیره پیام است.
4. **Database (DB)**
مخزن دائمی داده‌ها که پیام تماس را ذخیره می‌کند و پس از نوشتن، تائیدیه بازمی‌گرداند.

گام‌های پیام‌رسانی(Message Flow)

1. **RU → UI: fillContactForm(data)**
کاربر ثبت‌شده فرم تماس را با داده‌های مورد نیاز (name, email, message) پر می‌کند. این گام آغاز lifeline کاربر است و نشان‌دهنده ورود اطلاعات است.
2. **UI → CS: submitForm(data)**
WebUIداده‌های فرم را به ContactService ارسال می‌کند تا منطق پردازش و ذخیره‌سازی اجرا شود. این پیام معمولاً متد `submitForm()`را فراخوانی می‌کند.
3. **CS → DB: saveMessage(data)**
ContactServiceبرای ذخیره پیام تماس از طریق یک فراخوانی JDBC/ORM به پایگاه داده درخواست نوشتن می‌فرستد مثلاً .(`INSERT` SQL)

4. **DB → CS: confirmWrite()**
پس از انجام عملیات نوشتن، پایگاه داده یک پاسخ تائید (acknowledgment) به ContactService برمی‌گرداند تا مطمئن شود داده به‌درستی ذخیره شده است.
5. **CS → UI: returnStatus(success)**
ContactServiceنتیجه عملیات (موفق یا ناموفق) را به WebUI بازمی‌گرداند تا رابط کاربری بتواند به کاربر اطلاع دهد. این پیام وضعیت نهایی پردازش را نشان می‌دهد.
6. **UI → RU: displayConfirmation()**
WebUIبر اساس وضعیت دریافتی از سرویس، پیغام تائید (مثلاً "پیام شما ارسال شد") یا خطا را به RegisteredUserنمایش می‌دهد و lifeline کاربر را خاتمه می‌دهد

۵/۳ نمودار ماشین حالت(State Machine Diagram)

۱ .حالت‌های اصلی(States)

- **Draft**:حالت آغازین که محتوا در قالب پیش‌نویس ذخیره شده و قابل ویرایش است.
- **Review**:حالت بازبینی که محتوا توسط داور یا مدیر مورد ارزیابی قرار می‌گیرد.
- **Published**:حالت انتشار که محتوا پس از تأیید در دسترس عموم قرار می‌گیرد.
- **Archived**:حالت بایگانی که محتوا پس از پایان دورهٔ استفاده رسمی به آرشیو منتقل می‌شود و دیگر در لیست‌های فعال نمایش داده نمی‌شود.
- **[*]**:نماد شروع و پایان نمودار است؛ ورود به Draft آغاز و خروج از Archived به معنای خاتمهٔ چرخه است.

۲ .انتقال‌ها(Transitions)

- **Draft → Review : submitForReview()**
وقتی نویسنده پیش‌نویس را برای بازبینی ارسال می‌کند، رویداد `submitForReview()`فعال می‌شود و حالت به Review تغییر می‌کند.
- **Review → Published : approve()**
در صورت تأیید محتوای بازبینی‌شده، رویداد `approve()` رخ داده و محتوا منتشر می‌شود.(Published)
- **Review → Draft : requestChanges()**
اگر داور درخواست اصلاحات کند، رویداد `requestChanges()`فعال شده و محتوا به Draft بازمی‌گردد تا ویرایش شود.
- **Published → Archived : archive()**
پس از پایان دورهٔ نمایش یا به دلایلی مانند منسوخ‌شدن، رویداد `archive()` محتوا را به حالت Archived منتقل می‌کند.
- **Archived → [*]**
ورود به نماد پایان (`[*]`) نشان‌دهنده اتمام چرخهٔ زندگی محتوا است؛ دیگر انتقالی پس از آرشیو صورت نمی‌گیرد.