



Document de bones pràctiques a OutSystems

Revisió	Redactat per	Revisat per	Aprovat per	Data aprovació	Data publicació
1.0	Accenture	Arquitectura - CTTI			

1.1.1 Registre de canvis del document

Revisió	Apartat	Data	Redactat per	Canvis
1.0		11/07/2023	Accenture	Primera versió del document

RESPONSABLE DEL DOCUMENT=: Arquitectura – CTTI – LOT22


ARQUITECTE CTTI: Raúl García-Asenjo Prieto




Document de bones pràctiques a OutSystems

ÍNDEX

1.1.1	Registre de canvis del document	1
2.	Introducció	4
2.1	Objectiu	4
2.2	Referències	4
3.	Estructura d'aplicacions	5
3.1	Components generats	5
3.2	Biblioteques d'execució	5
3.3	Extensions de back-end personalitzades	6
4.	Pràctiques recomanades de Outsystems	7
4.1	Nomenclatura	7
4.1.1	Aplicacions	7
4.1.2	Mòduls	7
4.1.3	Entitats	9
4.1.4	Entitats Estàtiques	9
4.1.5	Atributs	9
4.1.6	Claus Forànies	9
4.1.7	Agregats i Consultes Avançades	10
4.1.8	Data Actions	10
4.1.9	Variables	10
4.1.10	Pantalles	11
4.1.11	Accions d'usuari	11
4.1.12	UI Flows	11
4.1.13	Interfície Widgets	11
4.1.14	Estructures	11
4.1.15	Site Properties	11
4.1.16	Web Blocks and Events	12
4.1.17	JavaScript, HTML i CSS	12
4.2	Patrons de disseny	12
4.2.1	Models de dades	12
4.2.2	Interfícies i regles	13
4.2.3	Seguretat	14
5.	Seguretat i rols dels usuaris	17
5.1	Administrar usuaris finals i organitzar Rols mitjançant grups	17
5.1.1	Crear un nou grup	17
5.1.2	Assignar Rols a un grup	17
5.1.3	Assignar Membres a un grup	17
5.1.4	Treure usuaris d'un grup	17
5.2	Rols d'usuari	18
5.2.1	Rols del sistema i Rols personalitzats	18
5.2.2	Persistència als Rols	18
5.2.3	Rols d'usuari al metamodel d'Outsystems	19
6.	Integracions amb sistemes de tercers	21
6.1	BBDD Integracions	21
6.2	Integració amb APIs	22
6.3	Integració amb serveis web	23
6.3.1	Consum de serveis web	23
6.3.2	Exposar serveis web	23
6.3.3	Depuració, supervisió i solució de problemes	23
6.3.4	Integració amb plataformes comercials	23
7.	Directrius de prova d'Outsystems	25

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>		N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems		
	N. versió aplicació: <i>1</i>	Build: 1	Pàg.3/30

7.1	Proves al llarg del cicle de vida de lliurament	25
7.2	Quines proves automatitzar	27
7.3	Segregació de proves Automatitzades	29

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1


2. Introducció

2.1 Objectiu

Aquest document s'adreça a totes les persones i organitzacions de CTTI que treballin en aplicacions o projectes amb l'eina d'OutSystems.

2.2 Referències

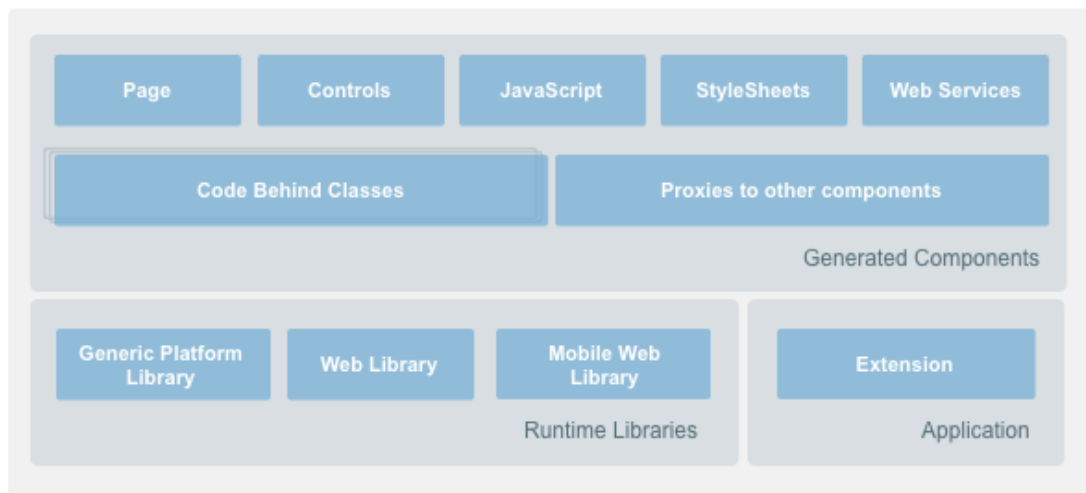
Referència	contingut
https://success.outsystems.com/documentation/best_practices	Best Practices documentació oficial OutSystems
https://www.outsystems.com/evaluation-guide/	Guia oficial d'OutSystems que respon a les preguntes més comunes dels equips
https://www.outsystems.com/forge/	Repositori de mòduls de codi obert, connectors, components d'interfície d'usuari i solucions comercials reutilitzables per ajudar a accelerar el temps de lliurament de les aplicacions d'OutSystems.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

3. Estructura d'aplicacions

Les aplicacions creades amb OutSystems inclouen tres tipus de components: generats, en temps d'execució i extensions de back-end personalitzades.

El diagrama següent mostra com es divideix una aplicació en els seus components subjacents:



Aquest document s'ha basat en la plantilla publicada al MQS Descripció de l'Arquitectura de la Solució v2.1

3.1 Components generats


Aquests components es creen automàticament cada vegada que s'envia una aplicació per a la seva compilació i publicació. Els components generats inclouen:

- Els components de la interfície d'usuari necessaris per admetre les pantalles mòbils i web o els blocs que dissenyi. Aquests estan fortament basats en JavaScript amb HTML5 i CSS3 estàndard.
- Les API REST i els serveis web SOAP es generen i exposen directament com a punts finals de back-end. També es generen els contractes corresponents (fitxers .json Swagger o .wsdl). A més, la plataforma crea automàticament un conjunt d'API REST com a serveis de back-end per a les aplicacions mòbils.
- La lògica de back-end/servidor que admet les seves regles comercials, la lògica de treball en segon pla i les API es generen com a codi de servidor optimitzat. Els servidors intermediaris a altres aplicacions a què es fa referència s'inclouen a la lògica de back-end com a assemblatges. Aquesta abstracció significa que les aplicacions que reutilitzen els serveis no es veuen afectades quan canvia la implementació del servei.

3.2 Biblioteques d'execució

Hi ha un conjunt de components inclosos a cada aplicació que admeten la funcionalitat bàsica, incloses les capacitats de representació bàsica i l'accés a diversos serveis dins de la plataforma OutSystems:

- **Biblioteca de plataforma genèrica:** Classes i mètodes del costat del client o del servidor utilitzats en totes les aplicacions generades, com les funcions integrades d'OutSystems disponibles a Service Studio, les utilitats d'administració de connexions de bases de dades, l'administració de llibres de treball

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: <i>1</i>

d'Excel, la administració de propietats del lloc, les utilitats del costat del servidor Ajax o el temporitzador utilitats relacionades.


- **Biblioteca mòbil i web:** controls Web/Javascript amb funcionalitat modularitzada per a cada giny base de la interfície d'usuari utilitzat per compondre les vostres interfícies d'usuari (per exemple, un component que implementa tota la funcionalitat relacionada amb una entrada, ja protegida per a la injecció de seqüències d'ordres) .

3.3 Extensions de back-end personalitzades

Assemblats a Integration Studio, aquests components personalitzats amplien OutSystems i us permeten incloure codi personalitzat en les vostres aplicacions.

Aquests components s'utilitzen en el context de qualsevol aplicació que hi faci referència i el codi s'executa al servidor front-end on s'implementa l'aplicació.

Contenen els assemblats implementats directament compilats o inclosos mitjançant Integration Studio, juntament amb els recursos addicionals de l'extensió.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

4. Pràctiques recomanades de Outsystems

4.1 Nomenclatura

Les següents són les pautes generals recomanades per anomenar objectes. També es preveu que hi pot haver algunes excepcions.

4.1.1 Aplicacions

- Nom curt i descriptiu.
- Incloeu sempre una descripció per donar més detalls.
- Descriu breument l'aplicació i les principals relacions i dependències externes.

Exemple:

Nom: RAC Reclamacions

Descripció: Reclamacions del Servei d'Atenció de Clients

4.1.2 Mòduls


4.1.2.1 Prefix Rules

- Utilitzeu prefixos per agrupar mòduls pertanyents a la mateixa aplicació i evitar conflictes de noms amb noms genèrics que es podrien utilitzar en altres aplicacions, com ara "Products" o "Members":
- Exemple: RAC_Products_CS, aplicació solar que conté tots els productes solars.
- Tots els mòduls que pertanyen a l'aplicació Reclamacions comencen amb RAC.
- Per als mòduls de la Fundació que gestionen els rols de les aplicacions (_Roles), aquesta convenció de prefixos no s'aplica. Exemple: ReclamacionsApprove_Roles.
- No utilitzeu prefixos als mòduls principals o de nucli que puguin ser reutilitzats estratègicament per altres aplicacions: Exemple: UserExtend_CS, mòdul que estén l'entitat Usuaris per emmagatzemar més informació de l'usuari.
- No utilitzeu prefixos als mòduls que serveixen com a punts d'entrada principals de les aplicacions per assegurar URL amigables: Exemple: UserManagement, mòdul que conté el portal per gestionar usuaris a OutSystems.

4.1.2.2 Suffix Rules

Mòduls de Foundation: Segons el contingut del mòdul, apliqui un dels següents sufixos:

- **_IS** - Serveis d'Integració: embolcall tècnic per consumir i normalitzar un servei extern.
- **_Lib** - Mòdul de Biblioteca Genèric.
- **_Th*** - Elements de Tema i Aparència Visual.
- **_Pat*** - Patrons d'IU reutilitzables només pel disseny, sense lògica empresarial.
- **_Rols** - Mòdul que assigna IAM FS a rols d'usuari OutSystems.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

Heu de contenir tots els rols que s'assignaran a IAM FS per a una aplicació comercial determinada. Aquest mòdul pot ser referenciat per altres mòduls de l'aplicació comercial per realitzar accions de validació d'accés d'usuaris. Per a mòduls específics d'aplicacions mòbils, afegiu una "M" abans de la convenció de noms, per exemple: `_Mth`


Mòduls de Serveis Principals de Negoci: Segons el contingut del mòdul, apliqueu un dels següents sufixos:

- `_CS`– Core Service. On es troba el model de dades i les accions CRUD.
- `_BL`- Lògica Empresarial Reutilitzable (Accions). On es troben totes les Server Actions i la Lògica de negoci
- `_Api`- API Externa: embolcall tècnic REST/SOAP per exposar serveis principals a consumidors externs a través de mecanismes d'integració.
Per a mòduls específics d'aplicacions mòbils, afegiu una "M" abans de la convenció de noms, per exemple: `_MCS`, `_MBL` o `_MCW`.
Mòduls d'usuari final i orquestració: Aquests mòduls no requereixen una convenció per assegurar URL amigables.
- `_CW`– Core Widgets. Components reutilitzables en diferents interfícies.



A més:

- Embolicar extensions en mòduls.
- Eviteu les referències circulars entre mòduls.
- Definir clarament les responsabilitats de cada mòdul.
- Elimineu les referències no utilitzades.
- No aïlleu el model de dades en un mòdul. Els mòduls han de ser unitats funcionals en comptes de les unitats arquitectòniques.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

4.1.3 Entitats

- Les convencions de noms són importants només per la consistència. Un pot triar codificar de manera diferent sempre que es mantingui la consistència de la convenció a tot el codi.
- Noms de clau primària d'entitat: Entitat "Country" = "CountryId" o simplement "Id".
- Noms importants a Pascal, en singular i sense espais entre les paraules (per exemple: "Contact", "ContactType").
- Eviteu tenir centenars d'atributs en una entitat
- Eviteu atributs grans (per exemple, atributs de text amb una longitud superior a 2000)
- Comproveu la regla d'eliminació de claus externes d'una entitat.
- Recordeu establir la propietat Is Mandatory (És obligatori).
- Afegiu descripcions almenys a les entitats.

4.1.4 Entitats Estàtiques


- Els noms de les Static Entities han de representar allò que realment són (per exemple, "ContactType").
- Noms importants a Pascal, en singular i sense espais entre les paraules (per exemple: "Contact", "ContactType").
- L'atribut "Id" a les Entitats estàtiques ha de ser tipus "text"

4.1.5 Atributs

- Atributs importants a Pascal, en singular i sense espais entre les paraules (per exemple: "PhoneNumber").

4.1.6 Claus Forànies

- Claus foranes en el format <NomCompletEntitat>Id o, si n'hi ha diverses, <Propòsit><NomCompletEntitat>Id (per exemple: "ContactTypeId" o "PrimaryContactTypeId").

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

4.1.7 Agregats i Consultes Avançades

Consultes:

- Utilitza agregats: estan optimitzats i són independents de la base de dades!
- Eviteu utilitzar índexs ([i]) en iterar un resultat de consulta.
- Minimitzeu el nombre de consultes que heu d'executar.
- Eviteu les conversions de tipus a Agregats.

Consultes Avançades:


- Utilitzeu consultes avançades només quan sigui absolutament necessari.
- Utilitzeu Parameters en lloc de valors codificats de forma rígida en SQL.
- Aplicar sagnia al codi SQL de manera coherent.
- Utilitzeu comentaris i comentaris en línia al vostre SQL.
- Utilitzeu consultes avançades per a operacions massives.
- No poseu la lògica empresarial en SQL.
- Utilitzeu la funció integrada d'EncodeSQL per a paràmetres en línia.
- Tingueu en compte que les consultes avançades són específiques de la base de dades.

4.1.8 Data Actions

Com a bona pràctica per a consultes excessives de dades utilitzeu les data actions per realitzar una sola trucada al servidor que recuperi totes les dades dels consultes utilitzats en aquesta data action

4.1.9 Variables

- Noms de variables a Pascal case, en singular o plural depenent de la informació emmagatzemada (per exemple: "ContactNumber", "MyIds").
- El nom de la variable d'identificador d'entitat ha de seguir el nom de l'entitat més "Id" (per exemple: "ContactId").
- El nom de la variable de registre ha de seguir el nom de la definició de registre més "Rec" (per exemple: "ContactRec").
- El nom de la variable de llista de registres ha de seguir el nom de la definició de registre més "List" (per exemple: "ContactRecList").

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

4.1.10 Pantalles

- Noms de pantalles (opcional) utilitzant el nom singular de l'<Objecte>, amb "" separant les paraules, i la <FuncionalitatPantalla> al final (per exemple: "Contact_Search").

4.1.11 Accions d'usuari

- *Noms d'Accions d'Usuari (opcional) utilitzant el nom singular de l'<Objecte>, amb "" separant les paraules, i la <FuncionalitatAccio> al final (per exemple: "Contact_Delete").*
- Les accions d'usuari associades a temporitzadors han de començar amb <Timer_>/<Bootstrap_> seguit de l'operació que es realitzarà (per exemple: "Timer_NotifyUsers").
- Les Accions d'Usuari (opcional) associades amb validacions de lògica empresarial han d'utilitzar el prefix Validate. Exemple: ("ValidateContact").

4.1.12 UI Flows

- UI Flows (opcional), anomenar utilitzant el <Propòsit o Perfil> seguit de "Flow" (per exemple, "ContactFlow" o "DocumentFlow").

4.1.13 Interfície Widgets


- Interface Widgets (opcional), anomenar utilitzant l'<Objecte> seguit del <Propòsit> a Pascal case (per exemple, "ContactEdit").

4.1.14 Estructures

- Structures (Estructures), el nom ha de fer servir un prefix per agrupar-les de manera lògica (per exemple, "st_Contact") i també per distingir les estructures personalitzades de les Estructures de Servei (creades automàticament) que tenen el mateix nom.
- Extension Structures (opcional), heu d'usar un prefix amb el nom de l'extensió (per exemple, "LDAP_User").

4.1.15 Site Properties

- Site Properties (opcional), el nom ha de fer servir un prefix, quan sigui aplicable, per agrupar-los de manera lògica (per exemple, "Email_UserAccount") - V10. A V11, hi ha carpetes a tot arreu per agrupar lògicament els elements.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: <i>1</i>

4.1.16 Web Blocks and Events

- Els Web Blocks nomenar utilitzant el <Propòsit o Perfil> seguit de "_Wb" (per exemple, "Participant_Wb").
- Els Events anomenar-los utilitzant el <Propòsit o Perfil> seguit de "_Event" (per exemple, "Refresh_Event").

4.1.17 JavaScript, HTML i CSS


- Comenta el teu Javascript.
- Utilitzeu JavaScript entre navegadors.
- Evita estils CSS duplicats.
- Minimitzeu JavaScript i CSS sempre que sigui possible.

4.2 Patrons de disseny

En aquest apartat es detallen per punts tant les bones pràctiques recomanades com els coneixements adquirits.

4.2.1 Models de dades

- **Indexa les teves entitats:** Crear índexs per als atributs importants més utilitzats millorarà significativament el rendiment de les consultes de selecció, encara que amb un lleuger cost addicional en les operacions d'inserció i actualització.
- **Evita realitzar unions sobre servidors vinculats:** Les unions entre servidors són molt ineficients pel fet que l'entitat del servidor vinculat es carrega completament al servidor de base de dades local i després es realitza la unió. Les unions entre servidors poden ser acceptables si les entitats són petites i inevitables, però com a regla general, cal evitar les unions entre servidors tant com sigui possible.
- **Aïlla les dades de text i binaris grans:** Evita utilitzar camps de text amb més de 2000 caràcters; els camps de dades més grans de 2000 bytes es converteixen en un tipus de dades Text, que alhora són molt més lents de processar i disminueixen el rendiment en general. Com a regla general, aïlla els camps binaris i/o de text grans en entitats separades i només recupera'ls quan siguin estrictament necessaris.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

- **Eviteu tenir Entitats amb massa camps** (250 camps poden ser massa): Implementeu associacions 1-1 per dividir els camps de l'Entitat quan es torni gran. Sempre separeu les dades grans de text i binaris a Entitats separades. Convenció de documentació per a atributs comuns d'entitats:
- ❖ **CreatedOn (DateTime)**: registra la primera data de creació. S'utilitza per arxivar dades que es van crear fa X quantitat de temps.
- ❖ **LastUpdatedOn (DateTime)**: registra la darrera data d'actualització. S'utilitza per arxivar dades que no s'han actualitzat des de temps X.
- ❖ **IsActive (Boolean)**: permet desactivar registres immediatament perquè puguin ser purgats/arxivats.
- ❖ **CreatedBy (UserId)**: proporciona traçabilitat sobre qui va crear la fila.
- ❖ **LastUpdatedBy (UserId)**: proporciona traçabilitat sobre el darrer usuari que va actualitzar la fila.

4.2.2 Interfícies i regles

- **Simplificar les preparacions de pantalla**- El processament de la preparació de pantalla afecta l'experiència de l'usuari i, per tant, ha de ser simple i ràpid d'executar. Això és especialment important per a les pantalles web amb un alt volum d'accés, com les pàgines d'inici.
- **Evitar l'ús de dades de preparació en accions de pantalla**- No utilitzeu dades de preparació en accions de pantalla, ja que això augmenta el trànsit de xarxa entre el servidor i el navegador. Això augmenta l'ús de la xarxa i degrada el rendiment del processament de sol·licituds del servidor.

4.2.2.1 Nomenclatura Elements Interfície


Tots els elements de pantalla han de tenir un nom explícit per facilitar la generació de seqüències de prova per a proves de regressió automatitzades.

Les classes CSS s'han d'utilitzar als elements de pantalla necessaris per facilitar les proves de regressió automatitzades.

Tots els noms assignats als elements de pantalla han de ser a CamelCase amb el prefix corresponent i sense guions baixos (per exemple: inpTankLevel, cntProductDetails, btnSearch, etc.) (opcional).

Es poden seguir els prefixos següents per mantenir els IDs generats curts (opcional):

- Button - btn
- CheckBox - cb
- ComboBox - cmb
- Container - ctn
- Expression - exp

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

- Icon - ico
- Image - img
- InputTextBox - inp
- Label - lbl
- Link - lnk
- ListBox - lb
- ListRecords - lr
- RadioButton - rb
- TableRecords - tr
- Upload - upl
- WebBlock – wb

Aquest document s'ha basat en la plantilla publicada al MQS Descripció de l'Arquitectura de la Solució v2.1


4.2.3 Seguretat

La demanda de cert nivell de seguretat de programari, fins i tot per a aplicacions petites, està creixent. És per això que, abans d'aprofundir en les vulnerabilitats de seguretat del programari i les tècniques de mitigació, és important comprendre les preguntes de seguretat de programari més comunes, com ara:

- Què vol protegir la seguretat del programari? I quins són els models de seguretat més utilitzats?
- Què és el model OSI?
- Com pot una organització assegurar als clients/usuaris que les seves aplicacions són fiables?
- On podeu trobar o com podeu saber quines són les principals vulnerabilitats de programari?
- Des de la perspectiva del desenvolupador, quin és l'abast pel que fa a la seguretat?

4.2.3.1 La Seguretat del Programari i el Model CIA


El Triangle de Seguretat de la CIA és un model que representa els tres pilars bàsics de la seguretat de la informació dins una organització, confidencialitat, integritat i disponibilitat. Aquesta tríada es considera els components més crucials de la seguretat i es fan servir per guiar les polítiques que protegeixen la informació duna organització.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1



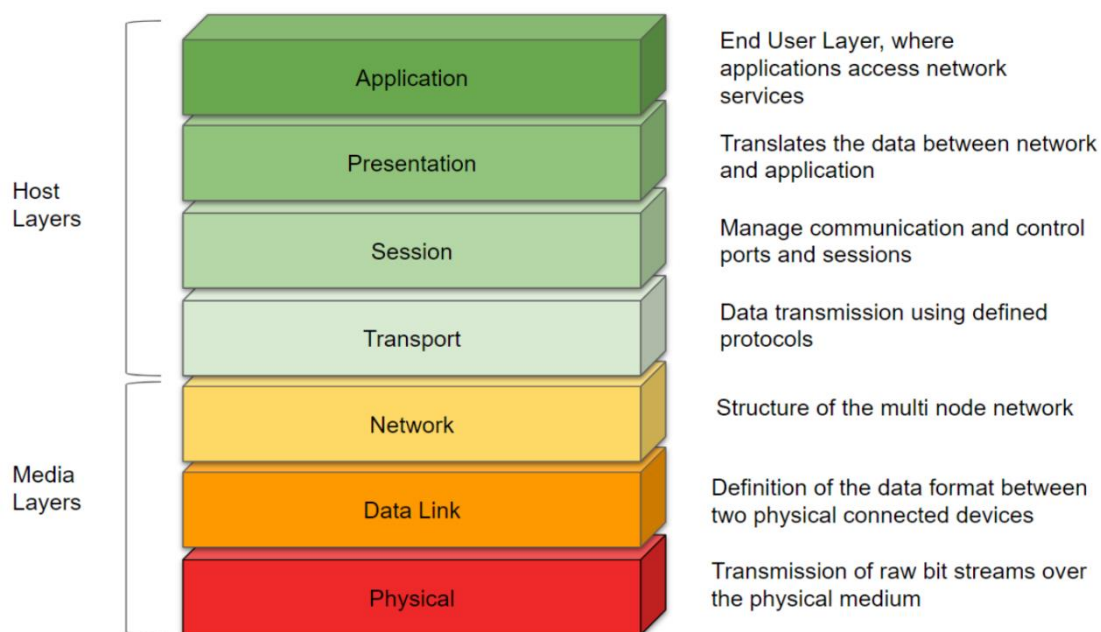
Comprovant-los detalladament:

- Confidencialitat** –La confidencialitat és la garantia que la informació és accessible només per a aquells autoritzats a tenir-hi accés. Podeu garantir la confidencialitat controlant l'accés a la informació, assegurant-vos que només aquells que estan autoritzats puguin accedir a la informació requerida. Això pot aconseguir seguint bones pràctiques d'autenticació, com ara contrasenyes segures, autenticacions multi factor, tokens de seguretat i certificats digitals. Heu de limitar l'accés a la informació confidencial i només aparèixer quan sigui necessari. També ha d'estar subjecte a la validació de rols. Els mitjans comuns utilitzats per administrar la confidencialitat inclouen llistes de control d'accés, volum, xifratge de fitxers i permisos de dades.
- Integritat:** integritat significa la confiança de les dades o recursos en termes de prevenció de canvis indeguts i no autoritzats. Les dades i els recursos no han de veure's compromesos ni manipulats, i han de ser fiables. Podeu verificar la integració de dades utilitzant sumes de comprovació com a pràctica. Si les dades van tenir modificacions, hi ha d'haver còpia de seguretat o redundància per garantir la recuperació de dades, cosa que també garanteix que quan una persona autoritzada realitza un canvi brut a les dades, pot revertir el dany.
- Disponibilitat:** la disponibilitat garanteix que els sistemes responsables de lliurar, emmagatzemar i processar la informació siguin accessibles cada vegada que els usuaris autoritzats els requereixin. Els sistemes d'alta disponibilitat (HA) són els recursos informàtics que tenen arquitectures dissenyades específicament per millorar la disponibilitat, amb maquinari i programari preparats per garantir que les dades confidencials estiguin sempre disponibles. La implementació d'arquitectures de sistema d'alta disponibilitat específiques pot evitar fallades de maquinari de destinació, actualitzacions o talls d'energia per admetre'n la disponibilitat, o podeu administrar diverses connexions de xarxa per encaminar diverses interrupcions de xarxa. Més recentment, es van afegir conceptes complementaris al model de la CIA, com ara:
- Autenticitat: fa referència a la característica de la comunicació, la documentació o qualsevol dada que garanteixi una qualitat genuïna.
- No repudieu: garanteix que el remitent d'un missatge no pugui negar més tard haver enviat el missatge i que el destinatari no pugui negar haver rebut el missatge.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

4.2.3.2 Model de responsabilitat compartida


El Model OSI és un model de responsabilitat compartida que serveix com a estàndard per definir la manera com els ordinadors es comuniquen entre si. Es divideix en set capes; Cada capa serveix a la capa superior i és servida per la capa inferior:



Aquest document s'ha basat en la plantilla publicada al MQS Descripció de l'Arquitectura de la Solució v2.1

La seguretat de les aplicacions desenvolupades en OutSystems és molt important. OutSystems treballa contínuament per proporcionar i millorar la protecció de seguretat incorporada a les capes de host, mitjançant l'aplicació de les últimes característiques de seguretat a la plataforma.

Aquesta és una de les principals raons per les quals ha de mantenir la seva plataforma OutSystems actualitzada a la darrera versió (Platform Server, LifeTime, Service Studio i Integration Studio).

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

5. Seguretat i rols dels usuaris

5.1 Administrar usuaris finals i organitzar Rols mitjançant grups

En administrar grans conjunts d'usuaris finals, pot ser difícil administrar els seus rols individualment. En lloc d'assignar rols manualment a cada usuari final, podeu crear grups d'usuaris finals i assignar els rols al grup.

Es configura des de: <https://?NomdelEntorn?/Users/>

5.1.1 Crear un nou grup

Per crear un nou grup:

1. A la pestanya Grups, feu clic a Crea un grup nou.
2. Assigneu un nom al grup i afegiu-hi una descripció.
3. Feu clic a Desa per crear el grup.

Quan creeu el grup, se us redirigirà a la pàgina de detalls del grup, on podeu afegir usuaris finals com a membres i assignar rols al grup.

5.1.2 Assignar Rols a un grup

Per assignar un rol al grup:

1. A la pàgina de detalls del grup, a la secció Rols, escriviu el rol requerit al quadre d'entrada.
2. Feu clic a Afegeix per assignar el rol al grup.

Podeu assignar tants rols com calgui al grup.

Quan s'afegeix un usuari final al grup, tots els rols que es concedeixen al grup es concedeixen a l'usuari. Aquests rols apareixeran a la pàgina de detalls de l'usuari amb el nom del grup sota la columna i no es poden treure individualment.

5.1.3 Assignar Membres a un grup

Per afegir un usuari final com a membre al grup:


1. A la pàgina de detalls del grup, a la llista d'usuaris finals de la secció Membres, escriviu el nom d'usuari necessari al quadre d'entrada.
2. Feu clic a Afegeix per afegir l'usuari final com a membre al grup.

Podeu afegir tants usuaris finals com calgui al grup.

Els grups d'un usuari final també es poden administrar a la pàgina de detalls de l'usuari, on podeu veure una llista de tots els grups a què pertany l'usuari i afegir nous grups o eliminar-ne els existents.

5.1.4 Treure usuaris d'un grup

Per treure un usuari d'un grup:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

1. A la pàgina de detalls del grup, a la secció Membres, escriviu el nom d'usuari requerit al quadre de cerca.
2. Col·loqueu el cursor sobre l'entrada de l'usuari final a la llista i feu clic a Suprimeix per treure l'usuari final del grup.

En treure un usuari d'un grup, es revocaran tots els rols concedits a l'usuari per herència del grup. Encara podeu afegir manualment un rol que es va concedir a l'usuari per herència d'un grup. Això anul·la l'herència del grup, és a dir, si assigneu un rol a un usuari final tant a través d'un grup com individualment, el rol no es revocarà si l'usuari final s'elimina del grup. En aquest escenari, la columna de la secció rols de la pàgina de detalls de l'usuari és buida.

5.2 Rols d'usuari

Utilitzeu Rols per restringir o permetre que els usuaris finals accedeixin a pantalles i operacions específiques de la seva aplicació.

Establiu rols en temps de disseny. Podeu utilitzar-los en dissenyar la lògica de la vostra aplicació i podeu associar-los amb els elements següents:

- Pantalla
- Activitat humana

5.2.1 Rols del sistema i Rols personalitzats


En crear un nou mòdul a Service Studio, OutSystems us proporciona un conjunt predeterminat de rols del sistema, però se us permet definir els vostres propis rols personalitzats. OutSystems proporciona els següents rols del sistema:

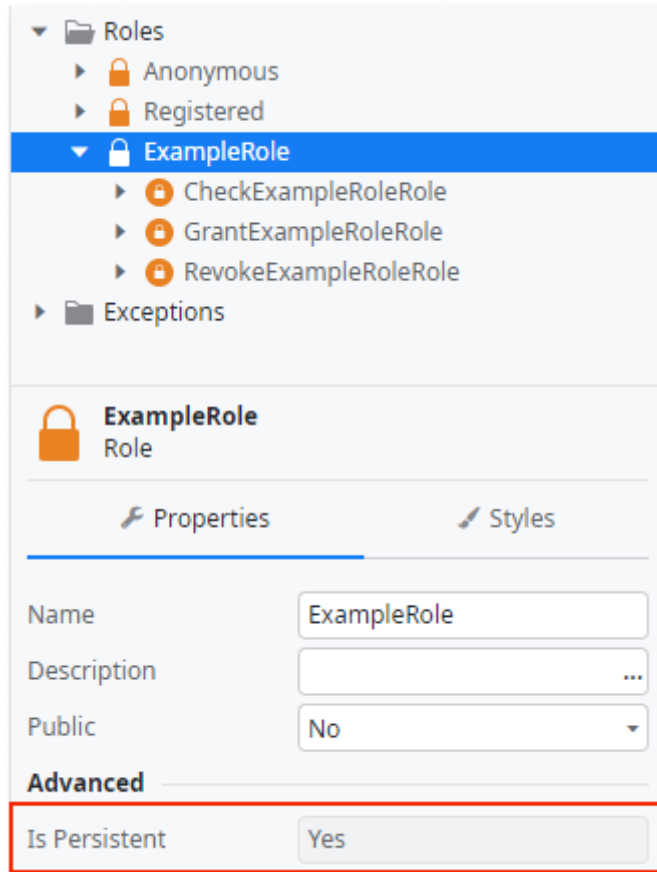
- **Anònim:** Permet que qualsevol usuari final accedeixi a l'element, inclosos els usuaris que no han iniciat sessió (usuaris no autenticats). Anònim és el rol més general i quan associa aquest rol, per exemple, amb una pantalla, tots els rols existents s'hi associen automàticament.
- **Registret:** Permet que qualsevol usuari final que hagi iniciat sessió en una aplicació que s'executa a la mateixa plataforma Server (usuaris autenticats) tingui accés a l'element. Això és possible gràcies al mecanisme d'inici de sessió únic d'OutSystems, que us permet compartir sessions d'usuari final entre aplicacions/mòduls. Quan associa aquest rol amb un element, tots els rols existents s'hi associen automàticament, excepte el rol anònim.

A més dels rols del sistema ja proporcionats, podeu definir els vostres propis rols personalitzats per administrar l'accés dels usuaris finals a les pantalles i el funcionament de la vostra aplicació.

5.2.2 Persistència als Rols

La concessió i revocació de rols durant el temps d'execució (mitjançant les accions Grant<Role name>Role i Revoke<Role name>Role) pot ser persistent en diverses sessions o només estar activa durant una sola sessió. Només podeu canviar aquesta configuració per a Aplicacions web tradicionals.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1



Aquest document s'ha basat en la plantilla publicada al MQS Descripció de l'Arquitectura de la Solució v2.1


- **Persistent:** El temps d'execució que atorga o revoca rols s'emmagatzema a la base de dades i es manté entre sessions d'inici de sessió. Establiu la propietat del rol a IsPersistent: Yes
- **No persistent:** El temps d'execució que concedeix o revoca rols no s'emmagatzema a la base de dades, i només dura una sessió. Quan tanqueu la sessió de l'usuari final, es perd el temps d'execució que atorga o revoca els rols. Establiu la propietat del rol a IsPersistent: No

5.2.3 Rols d'usuari al metamodel d'Outsystems

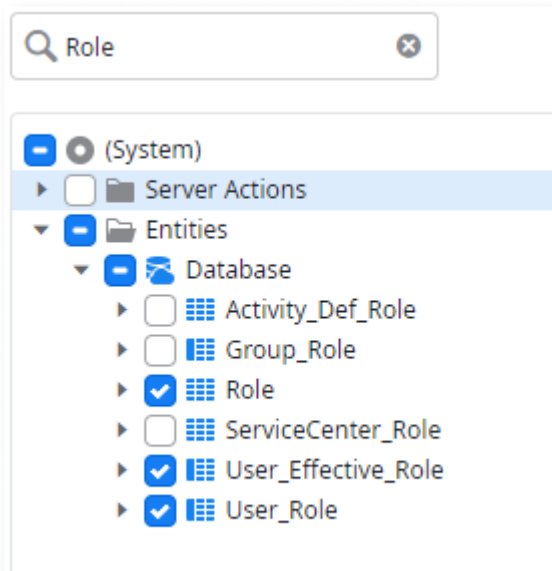
L'entitat Role emmagatzema els rols creats a la plataforma OutSystems. Com a desenvolupador, podeu verificar els rols d'un usuari en dues entitats diferents:


- L'entitat User_Role és una taula de meta model que emmagatzema l'associació d'usuari i el rol. Aquesta entitat només emmagatzema rols que s'afegeixen directament a un usuari específic mitjançant l'aplicació Usuaris.
- L'entitat User_Effective_Role és una vista de meta model que conté rols específics de l'usuari. Aquesta entitat inclou Rols assignats directament a un usuari i Rols assignats a un Grup a què pertany l'usuari.

Abans d'utilitzar User_Role i User_Effective_Role a la vostra aplicació, afegiu-los com a dependències seguint aquests passos:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

1. Obriu Administrar dependències.
2. A la llista de productors, seleccioneu (Sistema).
3. A la llista d'ítems públics, seleccioneu User_Role i User_Effective_Role.



 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

6. Integracions amb sistemes de tercers

Les següents són les integracions més freqüents que es fan servir comunament dins de la plataforma.

6.1 BBDD Integracions

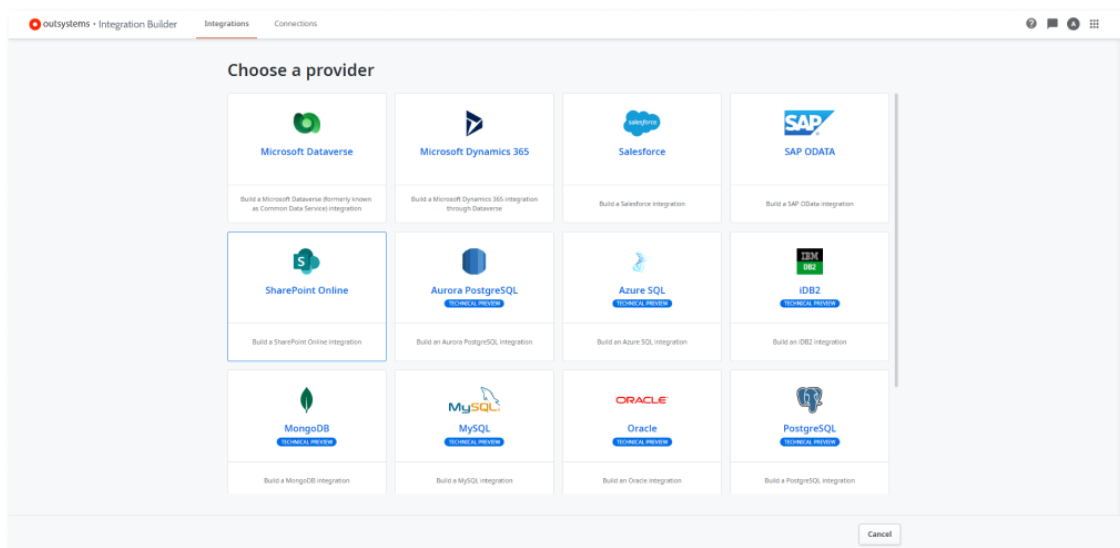
Podeu utilitzar Integration Builder per integrar les vostres aplicacions amb les bases de dades externes següents:

Bases de dades relacionals:

- DB2 iSeries
- PostgreSQL, Aurora PostgreSQL i Azure PostgreSQL
- MySQL
- Oracle
- Azure SQL i SQL Server


Bases de dades no relacionals:

- MongoDB

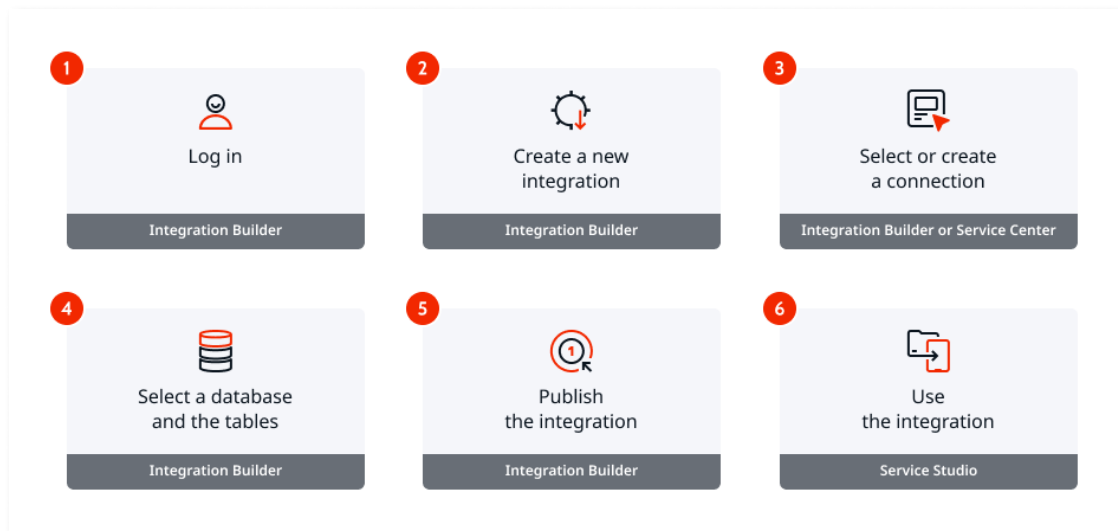


Per realitzar la integració amb una base de dades externa utilitzant Integration Builder, seguiu aquests passos:

1. Inicieu sessió a Integration Builder utilitzant l'URL del vostre entorn de desenvolupament i les credencials d'usuari de TI (nom d'usuari i contrasenya).
2. Creeu una nova integració per a un sistema extern i configureu els paràmetres.
3. Seleccioneu una connexió o creeu-ne una de nova per permetre que la vostra integració accedeixi a les dades a la base de dades externa.
4. Seleccioneu les taules o col·leccions que voleu que consumeixi la integració.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

5. Publiqueu la integració al vostre entorn.
6. Utilitzeu la integració a la seva aplicació.



El platform permet connectar-se a diferents databases simultàniament, identificant-les amb un nom que es farà servir en la definició dels magatzems de dades.

6.2 Integració amb APIs

Podeu integrar les vostres aplicacions d'OutSystems amb API REST proporcionades per altres sistemes, o fins i tot per altres aplicacions d'OutSystems. Utilitzeu aquesta funcionalitat per obtenir dades d'aquests sistemes o per sol·licitar-los que facin alguna acció.


En importar l'API REST, Service Studio fa el següent:

- Es connecta al servei web i analitza els mètodes i les estructures
- Crea mètodes API REST amb els paràmetres d'entrada i de sortida corresponents
- Crea les estructures per contenir els paràmetres d'entrada i de sortida corresponents sota un nou element d'arbre amb el nom d'API REST
- Mapeja els tipus de dades REST en tipus de dades de OutSystems

OutSystems tradueix els mètodes exposats per una API REST en accions d'OutSystems, amb la mateixa semàntica que qualsevol acció creada a Service Studio. Des de la perspectiva d'un desenvolupador, no hi ha cap diferència entre invocar un mètode d'OutSystems o un mètode exposat per un servei extern.

En consumir una API REST, podeu afegir lògica per personalitzar la informació que s'envia a les sol·licituds o es rep a les respostes. Les personalitzacions simples estan disponibles a Service Studio per modificar la informació de la sol·licitud original (com l'URL, el text de la sol·licitud o les capçaleres) o la informació de la resposta original (com el codi d'estat o el text de la resposta) .

Un cop s'ha integrat una API REST a la seva aplicació, l'API consumida es pot utilitzar en totes les aplicacions d'OutSystems dins del mateix entorn per recuperar o manipular informació.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: <i>1</i>

6.3 Integració amb serveis web

Els serveis web SOAP i les API REST són simples amb OutSystems. Els equips de desenvolupadors els poden consumir i exposar sense problemes sense haver d'escriure codi. La depuració, la monitorització i la solució de problemes estan integrats. L'editor visual genera artefactes com a mètodes i estructures de dades, que els desenvolupadors poden utilitzar a les aplicacions de forma visual.

6.3.1 Consum de serveis web

Per consumir un servei web SOAP, els desenvolupadors només proporcionen la ubicació del fitxer WSDL (llenguatge de descripció de serveis web). L'IDE d'OutSystems inspecciona el WSDL i genera tot allò necessari per invocar els mètodes del servei web.

Per consumir una API REST, un desenvolupador ha de proporcionar l'URL del punt final del servei i un exemple de la sol·licitud i resposta del servei al JSON. La majoria dels serveis inclouen una sol·licitud i resposta de mostra JSON a la seva documentació i es poden enganxar a l'editor d'OutSystems. OutSystems genera tot allò necessari per invocar el servei REST.

Després que OutSystems hagi generat els mètodes de servei i les estructures de dades, la invocació del servei és perfecta. Des de la perspectiva d'un desenvolupador, no hi ha cap diferència entre invocar un mètode del sistema o un mètode exposat per un servei extern.

6.3.2 Exposar serveis web

Amb OutSystems, els desenvolupadors poden exposar qualsevol part de la lògica de l'aplicació com un servei web. Tot això es fa a l'editor visual. La definició del servei, com a mètodes i estructures de dades, juntament amb la lògica del servei, es defineixen visualment.

Els serveis web són una part integral de les aplicacions desenvolupades. Quan un equip implementa una aplicació, tots els punts finals dels serveis web es creen automàticament i el servei està llest per utilitzar-se. No hi ha cap necessitat de configuracions o implementacions addicionals.


En exposar les API REST, OutSystems genera automàticament la documentació i la posa a disposició al punt final del servei. Aquesta documentació es pot personalitzar completament.

6.3.3 Depuració, supervisió i solució de problemes

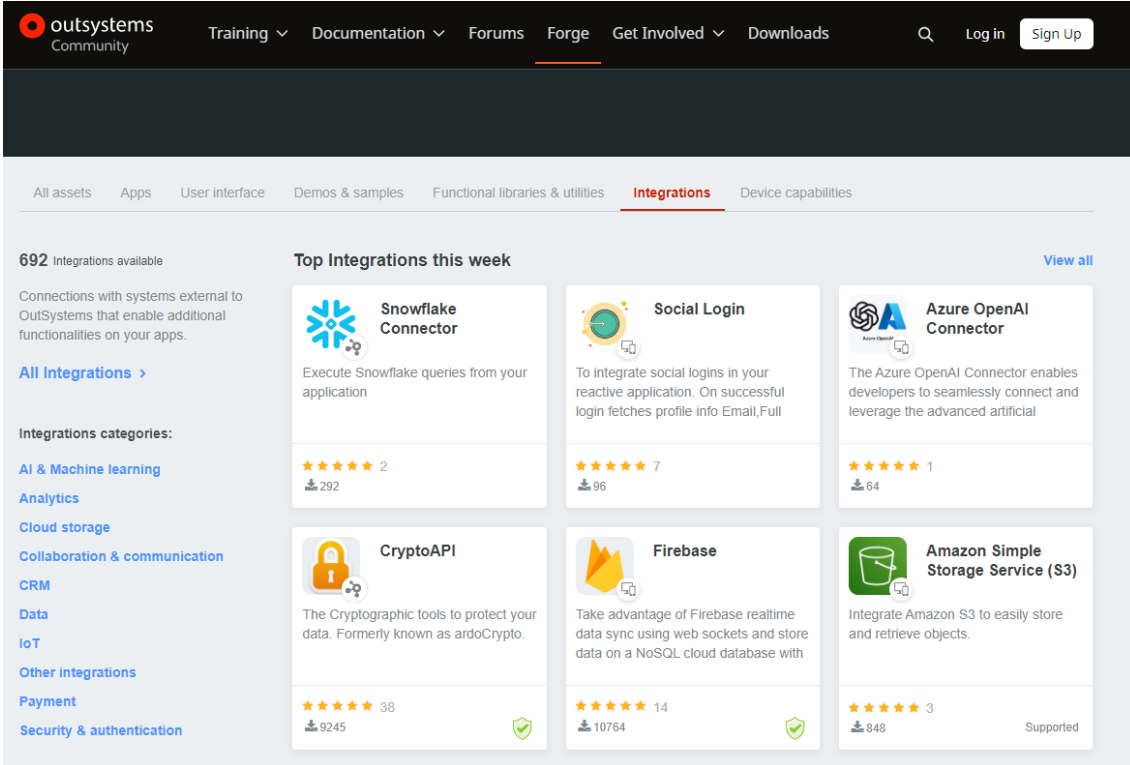
Els desenvolupadors poden depurar una integració de serveis a l'editor visual. Si els desenvolupadors consumeixen un servei, poden inspeccionar els valors enviats al servidor per veure què torna el servei. Si els desenvolupadors exposen un servei, poden passar a la implementació del servei.

OutSystems instrumenta automàticament totes les aplicacions i la integració. En temps d'execució, es recopilen errors, auditories i dades de rendiment, cosa que permet comprovar si una integració té errors o un impacte negatiu en el rendiment de l'aplicació.

6.3.4 Integració amb plataformes comercials


 <p>Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació</p>	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

OutSystems té un catàleg amb alguns connectors per interactuar amb les diferents plataformes comercials existents. Aquest catàleg es pot trobar a <https://www.outsystems.com/forge/>



The screenshot shows the 'Integrations' section of the OutSystems Forge. It features a navigation bar with 'outsystems Community' and various menu items like 'Training', 'Documentation', 'Forums', 'Forge', 'Get Involved', and 'Downloads'. Below the navigation, there are tabs for 'All assets', 'Apps', 'User interface', 'Demos & samples', 'Functional libraries & utilities', 'Integrations' (which is selected), and 'Device capabilities'. The main content area displays '692 Integrations available' and a 'Top Integrations this week' section. This section contains six integration cards, each with an icon, title, description, and user ratings. The cards are: Snowflake Connector, Social Login, Azure OpenAI Connector, CryptoAPI, Firebase, and Amazon Simple Storage Service (S3). Each card also shows the number of users and a star rating.

Aquest document s'ha basat en la plantilla publicada al MQS Descripció de l'Arquitectura de la Solució v2.1

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

7. Directrius de prova d'Outsystems

Les aplicacions creades amb la plataforma OutSystems es beneficien de la validació contínua de la integritat, que fa un seguiment de l'impacte dels canvis en tota l'aplicació (model de dades, lògica empresarial i presentació) i garanteix que no es trenqui res al moment de la implementació.

Les capacitats de recuperació automàtica corregeixen automàticament els problemes i notifiquen als desenvolupadors de qualsevol problema que hagin de manejar. Cada cop que hi ha un canvi, OutSystems fa un seguiment de totes les dependències, automatitza les actualitzacions de la base de dades i analitza l'impacte que el canvi tindrà a les aplicacions en execució al llarg del seu cicle de vida.


En un nivell més ampli, OutSystems fa anàlisis d'impacte per a múltiples aplicacions en crear plans d'implementació a LifeTime. Avaluat l'impacte de moure les versions d'aplicació seleccionades a l'entorn de destinació abans d'executar el pla d'implementació.

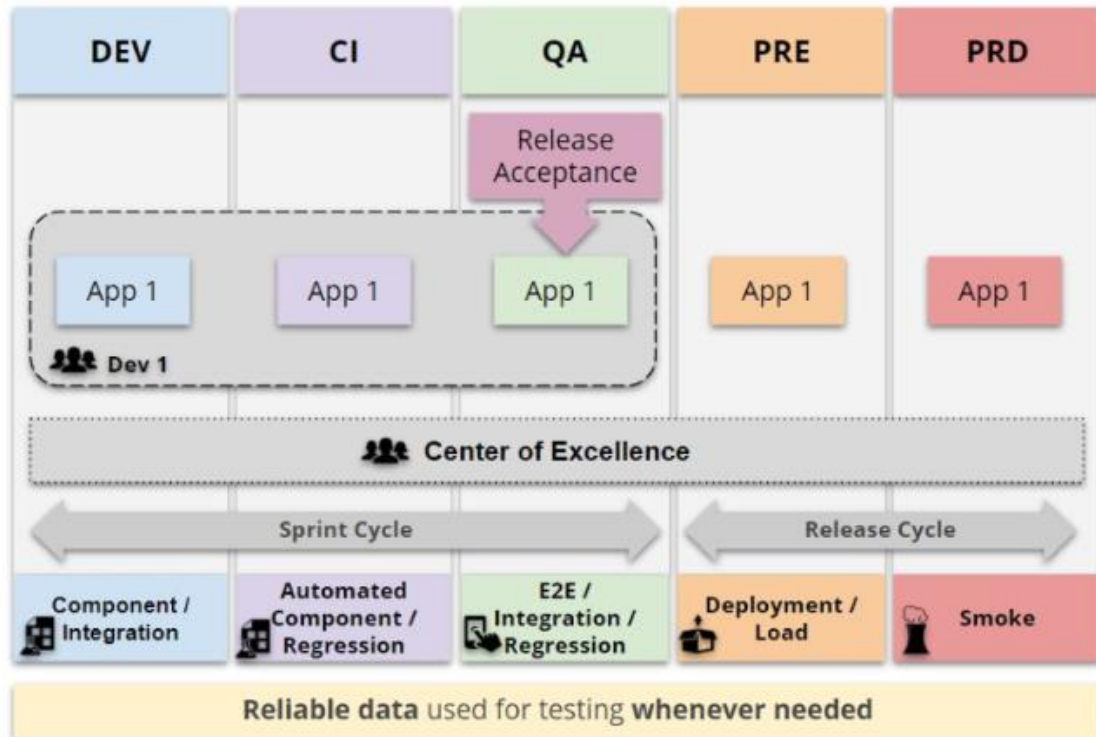
Aleshores, què obtens de les proves amb OutSystems? El nombre d'errors introduïts sol ser molt menor en comparació amb les tecnologies de desenvolupament tradicionals, la qual cosa es tradueix en menys cicles de prova i correcció i redueix en gran mesura l'esforç de desenvolupament i lliurament.

Tingueu en compte que les validacions d'integritat, les capacitats de recuperació automàtica i l'anàlisi d'impacte no eliminaran la necessitat de fer proves durant el cicle de vida de l'aplicació. OutSystems fa realment la integritat del paquet de programari, validant la qualitat general de l'aplicació, que inclou un conjunt de criteris funcionals, de rendiment i de seguretat.

7.1 Proves al llarg del cicle de vida de lliurament

El diagrama següent mostra les activitats de prova típiques que poden tenir lloc durant el cicle de vida de lliurament de programari de OutSystems:

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1



Aquest document s'ha basat en la plantilla publicada al MQS Descripció de l'Arquitectura de la Solució V2.1


Les proves de components validen el comportament de les unitats individuals de codi. Aquestes proves corresponen a proves unitàries en tecnologies tradicionals de codi alt. A OutSystems s'assignen a elements d'acció que implementen la lògica empresarial. Basat en la metodologia Agile com a part de les activitats de desenvolupament de sprints, els desenvolupadors han d'automatitzar o lliurar manualment aquestes proves a l'entorn DEV. Les proves automatitzades de components també s'han d'executar a l'entorn d'integració contínua (CI), com a part del flux de CI. Les proves de components poden adoptar un de dos enfocaments:

- Solidaritat, que incorpora integracions amb components externs de OutSystems.
- Solitari, que aïlla l'objectiu de la comunicació amb altres components d'OutSystems.

Les proves d'integració/API validen la integració amb sistemes externs, com ara una capa de middleware, o una integració directa amb qualsevol altre sistema independent, que també inclou un altre sistema feblement acoblat fet amb OutSystems, per exemple, aplicacions que exposen API a parts externes. Els desenvolupadors han d'automatitzar o executar proves manualment a DEV, o fer-ho QA en cas que les integracions de destinació no estiguin disponibles o no siguin fiables a DEV. Igual que amb les proves de components, algunes de les proves d'integració poden executar-se automàticament com a part del flux de CI; No obstant això, aquesta pràctica s'ha de limitar a les proves que s'executen ràpidament.

Les proves de sistema o extrem a extrem (E2E) validen la funcionalitat completa des de la perspectiva de l'usuari final o del sistema. Les proves d'extrem a extrem generalment s'executen a través d'una interfície d'usuari web o mòbil (proves d'interfície d'usuari), però no totes les proves E2E són proves d'interfície d'usuari. És possible que vulgueu començar automatitzant només els casos d'ús crítics, particularment els camins feliços. Per exemple, provar una API exposada per una aplicació OutSystems a parts externes es pot considerar una prova E2E, igual que el conjunt d'accions d'usuari del costat del servidor que comprenen un flux d'usuari final particular. Els desenvolupadors solen ser responsables d'automatitzar aquestes proves,

En general, les proves E2E no són part del flux de CI perquè tendeixen a ser lentes. Les proves E2E automatitzades es poden executar en QA, incloure's a la canalització de CI/CD o també executar-se manualment.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

Les proves de regressió validen que les característiques acceptades prèviament continuen funcionant com s'esperava en les noves versions d'una aplicació, i també que els canvis d'infraestructura no interrompen la funcionalitat existent. A mesura que creix la seva fàbrica de codi baix OutSystems, també ho fa la necessitat d'automatitzar les proves de regressió. De fet, l'automatització és l'única manera d'accelerar la validació de canvis nous en les aplicacions existents. El lliurament continua depèn molt d'aquesta capacitat, ja que només pot lliurar canvis tan ràpid com pugui validar-ne l'impacte.

Les proves de rendiment avaluen la capacitat de resposta i escalabilitat de les aplicacions.

Les proves de càrrega són les més comunes. Les proves de càrrega validen el comportament de l'aplicació en càrregues de treball predefinides (esperades davant de pics) en una varietat d'escenaris. Per a les aplicacions que consumeixen més dades, enlloc de proves amb concurrència d'usuaris, pot tenir més sentit provar el rendiment amb grans quantitats de dades. Les proves de rendiment generalment s'executen en un entorn PRE, encara que comprometen PRE perquè és un entorn PRD de referència quan les proves de càrrega triguen massa a executar-se. Idealment, hi hauria d'haver un entorn dedicat només per instanciar aquesta capacitat específica.

Les proves d'acceptació/funcionals validen que la funcionalitat nova compleix els requisits empresarials abans d'introduir nous canvis en producció. Les proves d'acceptació solen dirigir-se a les característiques crítiques per al negoci d'una versió, i normalment es fan manualment a l'entorn de control de qualitat abans del llançament. No obstant això, els desenvolupadors poden utilitzar proves d'acceptació a DEV per validar característiques abans de promoure-les a CI o QA.

Les proves de seguretat validen la seguretat d'un sistema determinat i la seva infraestructura subjacent. Una estratègia típica és explotar vulnerabilitats de seguretat comunes en temps d'execució. Les proves de seguretat s'han de fer com més aviat millor en el cicle de vida de lliurament (tendència shift-left), normalment per equips de seguretat, que poden ser interns o subcontractats com a servei. Realitzeu aquest tipus de proves en qualsevol dels entorns, des de DEV fins PRE-PROD i PROD on siguin crucials. Per exemple, les proves de seguretat d'injecció SQL i XSS es poden compilar i executar a DEV perquè l'equip de DEV pugui promoure ràpidament correccions. L'execució de proves de seguretat a DEV també ensenya als desenvolupadors com evitar la introducció de vulnerabilitats de seguretat comunes.


Les proves de fum estan dissenyades per validar ràpidament l'estabilitat i la funcionalitat bàsica de les compilacions de programari generades per l'equip d'operacions a PRE i PRD. Heu d'incloure passos de prova de fum, que ha de documentar al runbook. Aquest tipus de prova normalment s'executa manualment. Però quan el runbook s'executa automàticament, aquestes proves també es poden automatitzar.

Provar una aplicació sempre requereix la inversió de l'equip de lliurament. Per aquesta raó, el tipus d'activitats de prova i la quantitat d'esforç de prova invertit dependran en gran mesura de la complexitat o l'impacte comercial de l'aplicació que es lliura. Com a regla general, com més gran sigui el risc associat amb la funcionalitat de l'aplicació, més a fons l'ha de provar.

És molt recomanable utilitzar dades realistes sempre que sigui possible per donar suport a diferents activitats de prova. Això permet escenaris de prova realistes i ajuda a identificar problemes relacionats amb les dades més aviat al cicle de lliurament, abans d'arribar a producció. Per aconseguir-ho, els equips de lliurament han de comprendre les opcions que tenen per fer que cada prova sigui el més ràpida i idempotent possible. Llegiu Administració de dades de prova per obtenir més detalls.

7.2 Quines proves automatitzar

Sabem per l'Informe de l'Estat de DevOps que les empreses d'alt rendiment automatitzen la majoria de proves. Si teniu el lliurament continu en ment com a objectiu, heu d'esforçar-vos per automatitzar les proves de regressió tant com sigui possible.

 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: 1
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: 1	Build: 1

És important ser selectiu amb automatització. Tractar d'automatitzar tot poques vegades és una estratègia efectiva, ja que algunes funcionalitats es poden fer servir amb poca freqüència, tenir poc impacte si falla o ser costoses de provar.

Angie Jones, coneguda defensora sènior de desenvolupadors d'automatització, diu que l'automatització s'ha d'abordar amb compte, perquè per la seva naturalesa l'automatització requereix temps i manteniment i l'automatització redundant és sorollosa.

Mantenir scripts de prova automatitzats pot ser un gran esforç, per la qual cosa és important tenir en compte el valor de l'automatització en relació amb l'esforç requerit per mantenir-la. Penseu en les proves automatitzades com qualsevol altre programari que desenvolupi i mantingui.

Identifiqueu què voleu automatitzar i com ha de dissenyar la vostra prova perquè sigui realment eficient en termes de manteniment del codi i temps d'execució. Eviteu crear codi de prova redundant que no sigui reutilitzable.

Aleshores, com decideixes quines proves automatitzar? Bé, ens agrada l'enfocament que Angie Jones recomana:

1. Comenceu amb el vostre instint. De vegades hi ha alguna cosa que ens diu que una prova específica aportarà valor si la podem automatitzar. Respecteu la vostra intuïció en determinar quines proves automatitzar.
2. Avalueu el risc de no automatitzar una prova, tenint en compte la freqüència amb què s'utilitza la funcionalitat objectiu i l'impacte en el negoci si es trenca.
3. Avaluar el valor de la prova, basant-se en la distinció de la prova i també en el temps que s'hauria de recuperar d'un error de la funcionalitat sota prova
4. Avalueu la rendibilitat de construir la prova, analitzant què tan ràpid i fàcil és construir la prova.
5. Avaluar l'historial de la història d'usuari que volem provar, quants errors funcionals vam tenir en el passat en àrees similars contra el nombre d'errors d'aquest cas de prova en particular.
6. La puntuació per a cada ítem prové de multiplicar els dos valors determinats a cada vector. Al final, heu d'acabar amb una matriu de puntuació que l'ajudarà a decidir si ha d'automatitzar o no cada prova específica que necessita realitzar.
 - > **74**significa que realment hauries d'automatitzar la prova
 - > **24 i** < **75**vol dir que ha de decidir en funció de la intuïció.
 - < **25**no has d'automatitzar la prova perquè no aportarà el valor necessari

Per exemple:

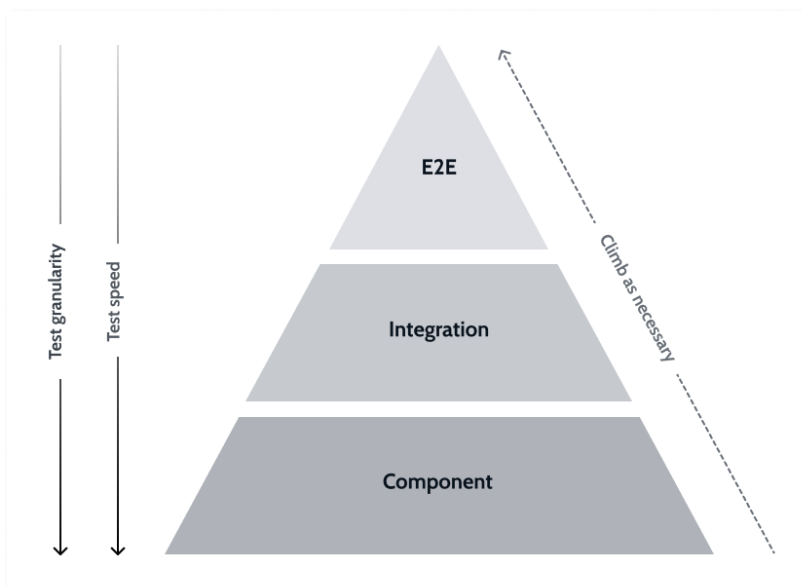



ID	Description	Risk	Value	Cost	History	SCORE
1	Create Case	25	25	25	0	75
2	Update Case	25	20	20	0	65
3	Close Case	5	5	10	0	20

Aquest document s'ha basat en la plantilla publicada al MQS
Descripció de l'Arquitectura de la Solució v2.1

7.3 Segregació de proves Automatitzades

L'automatització de proves redueix en gran mesura les proves de regressió en sistemes cada cop més complexos o de missió crítica. També és una part crítica de qualsevol enfocament de lliurament continu. En triar quines proves automatitzar, és molt recomanable adoptar l'enfocament de la piràmide de proves, que estableix que l'esforç d'automatització de proves s'ha de distribuir d'acord amb una forma de piràmide, amb proves de components a la capa inferior i movent-se fins a les proves E2E. La raó daixò és que les proves de components són fàcils descriure i mantenir, així com molt ràpides d'executar, mentre que les proves E2E solen ser inestables, difícils de mantenir i requereixen molt temps d'execució.



 Generalitat de Catalunya Centre de Telecomunicacions i Tecnologies de la Informació	<i>Lot 22 Arquitectura OutSystems</i>	N. revisió doc.: <i>1</i>
	Document de bones pràctiques a OutSystems	
	N. versió aplicació: <i>1</i>	Build: 1

Un concepte clau per implementar la piràmide de proves és escriure aplicacions comprovables. A OutSystems, això vol dir adoptar els principis d'arquitectura de 4-Layer Canvas i distribuir la funcionalitat en peces petites i comprovables. Per exemple, totes les entitats empresarials resideixen en mòduls de capa central i estan embolicades per accions de servidor públic que validen la correcció de dades, les dependències i els càlculs. No ha d'haver lògica empresarial en les accions de pantalla perquè no es poden provar amb proves de components. Les proves segregades permeten als equips de lliurament utilitzar proves d'interfície d'usuari automatitzades amb moderació per a fluxos d'interfície d'usuari crítics i escriure un conjunt complet de proves unitàries i d'integració fàcilment automatitza-les que cobriran la funcionalitat empresarial crítica i definiran l'abast de la regressió per a cada versió nova de l'aplicació.