VALTIOKONTTORIN JULKAISUJA | TIETOHALLINTO 6/2024

Tietotuotantoa tukeva mallipohjainen tietojärjestelmäkehitys

Hotti Virpi, Kallio Maija

Valtiokonttori
Statskontoret
State Treasury

Kuvailulehti

Julkaisija: Valtiokonttori	Julkaisupäivämäärä: 19.6.2024
Tekijät: Hotti Virpi, Kallio Maija	Julkaisun teema: Tietohallinto
Julkaisun nimi: Tietotuotantoa tukeva mallipohjainen tietojärjestelmäkehitys	ISSN/ISBN: 2489-4761 / 978-951-53-3890-7

Tiivistelmä:

Tietojärjestelmä on tarkoitettu tietojen keräämiseen, manipulointiin, tallentamiseen, jakamiseen ja hallinnointiin. Se voi koostua erilaisista komponenteista kuten pilvipalveluista, verkkosovelluksista ja tietokannoista, joita hallitaan ja valvotaan erilaisten mekanismien avulla.

Mallipohjaisuus tietojärjestelmien kehittämisessä tarkoittaa erilaisten mallielementtien kuvaamista sellaisella tavalla, että malli voidaan validoida sekä validoidusta mallista voidaan generoida käyttäjille vuorovaikutuksen mahdollistava osa (frontend) ja tietojen käsittelyn mahdollistava osa (backend). Kun mallinnus tehdään sellaisella työkalulla, joka mahdollistaa tarvittaessa käyttöliittymiin ja ohjelmistorajapintoihin tarvittavien komponenttien sisällyttämisen osaksi generoitavaa tietojärjestelmäkokonaisuutta, siirrytään tietojärjestelmien kehittämisessä uuteen aikakauteen.

Tässä julkaisussa kuvataan Suomen elpymis- ja palautumissuunnitelman toimeenpanon toteutukseen kehitetyn RRP-järjestelmän mallielementit ja havainnollistetaan niiden merkitystä tietojärjestelmäkehityksessä. Lisäksi kuvataan mukautuvan ja tehokkaan tietotuotannon mahdollistavien tietojärjestelmien kehitystapoja.

Asiasanat:	tietojärjestelmä, tietotuotanto, mallipohjaisuus, koodigenerointi		
Kieli:	suomi	Sivumäärä:	19
Julkaisujen myynti/jakelu:	valtiokonttori.fi/julkaisut		

Beskrivningsblad

Publicerare: Statskontoret	Publiceringsdatum: 19.6.2024
Utfärdare: Hotti Virpi, Kallio Maija	Publikationens tema: Informationsförvaltning
Publikationens namn: Modellbaserad informationssystemutveckling som stödjer informationsproduktion	ISSN/ISBN: 2489-4761 / 978-951-53-3890-7

Sammandrag:

Informationssystemet är avsett för att samla in, manipulera, lagra, dela och hantera data. Det kan olika komponenter som molntjänster, webbapplikationer och databaser, som hanteras och övervakas med olika mekanismer.

Modellbaserad i utvecklingen av informationssystem innebär att beskriva olika modellelement på ett sådant sätt att modellen kan valideras, och från den validerade modellen en del som möjliggör för användare att interagera (front-tend) och en del som möjliggör databehandling (backend) kan genereras. När modelleringen görs med ett verktyg som vid behov gör att de komponenter som behövs för användargränssnitt och applikationsprogrammeringsgränssnitt kan ingå som en del av den genererade informationssystemhelheten, går vi in i en ny era i utvecklingen av informationssystem.

Denna publikation beskriver modellelementen i RRP-systemet som utvecklats för genomförandet av Finlands återhämtning och återhämtningsplan och illustrerar deras betydelse för utvecklingen av informationssystem. Dessutom beskrivs utvecklingsmetoder för informationssystem som möjliggör flexibel och effektiv dataproduktion.

Ämnesord:	informationssystem, informationsproduktion, modellbaserat, kodgenerering		
Språk:	finska	Sidantal:	19
Publikationernas försäljning/utdelning:	valtiokonttori.fi/sv/publikationer		

Description sheet

Published by: State Treasury	Publication date: 19/06/2024
Authors: Hotti Virpi, Kallio Maija	Theme of publication: ICT Management
Title of publication: Model-based information system development supporting information production	ISSN: 2489-4761 / 978-951-53-3890-7

Abstract:

The information system is intended for collecting, manipulating, storing, sharing and managing data. It can have different components, such as cloud services, web applications and databases, which are managed and monitored using different mechanisms.

Model-based in the development of information systems means describing different model elements in such a way that the model can be validated, and from the validated model a part that enables users to interact (frontend) and a part that enables data processing (backend) can be generated. When the modeling is done with a tool that allows, if necessary, the components needed for user interfaces and application programming interfaces to be included as part of the generated information system entity, we move into a new era in the development of information systems.

This publication describes the model elements of the RRP system developed for the implementation of Finland recovery and resilience plan and illustrates their importance in information system development. In addition, the development methods of information systems enabling flexible and efficient data production are described.

Keywords:	information system, information production, model-based, code generation		
Language:	Finnish	Pages:	19
Publication sales/ distribution:	valtiokonttori.fi/en/publications		

Sisällysluettelo

1 Johdanto

Erityisesti julkisessa hallinnossa on useita tietojärjestelmiä, jotka ovat elinkaarensa päässä. Osa voi olla toteutettu Cobolilla tai jollakin muulla ohjelmointikielellä, joiden osaajia ei enää ole olemassa. Osaa elinkaarensa päässä olevista tietojärjestelmistä ei enää tarvita tai niissä olevat toiminnallisuudet voidaan hoitaa toisilla tietojärjestelmillä. On myös tilanteita, että yksi tai useampi tietojärjestelmä pitää modernisoida – modernisointi tarjoaa mahdollisuuden miettiä myös modernisointitapoja. Tässä dokumentissa käsitellään mallipohjaista kehittämistä, joka on yksi tapa modernisoida, ja joka mahdollistaa mallin validoinnin ja koodipohjan generoinnin.

Kun tietojärjestelmiä uudistetaan, ajantasaisten määrittelyjen (specifications) lähde on usein ohjelmistokoodi. Muu dokumentaatio on saattanut jäädä päivittämättä tai ajantasainen dokumentaatio on hajanaista. Mallipohjaisen työkaluista suurin osa tukee eteenpäin suuntautuvaa suunnittelua (model-driven engineering, MDE), ja harvat työkalut tukevat kooditasolta lähtevää käänteis- tai uudelleensuunnittelua (model-driven reverse or reengineering, MDRE) (Lano & Siala, 2023, 9.1[1]). Koodipohjan generoiminen tietojärjestelmien mallien pohjalta auttaa siis tältä osin teknisen velan (technical debt) vähentämisessä.

Laajojen kielimallien käyttäminen sekä eteenpäin että käänteis- tai uudelleensuunnittelussa on lisääntynyt vuodesta 2022. Kielimallien käyttäminen graafisten ja tekstuaalisten mallinnustyökalujen rinnalla voi nopeuttaa mallinnusta, ja näin on helpompi tukea sellaista iteratiivista mallinnusta, jossa käyttäjiltä pyritään saamaan palautetta mahdollisimman varhaisessa vaiheessa. Puhutaan monimuotoisesta (blended) mallinnuksesta, jossa korostetaan käyttäjäkokemusta ja mahdollisesti asetetaan käyttäjäkokemus järjestelmän oikeellisuuden edelle (David et al., 2023[2]).

Tässä dokumentissa tarkastellaan, mitä mallipohjainen kehittäminen on ketterässä viitekehyksessä (Luku 2). Luvussa 3 tarkastellaan, mitä asioita mallinnuksessa tulee huomioida ohjelmistotuotannon näkökulmasta. Dokumentissa tuodaan esille, miten LeBlanc Designer -mallinnustyökalua (Luku 4) käytettiin, kun Valtiokonttorille annettiin syksyllä 2021 tehtäväksi kehittää uusi tietojärjestelmä EU:lle raportoitavien tietojen keräämiseksi. Koska järjestelmän tekoaikaa oli vain muutama kuukausi ja järjestelmän elinkaari muutama vuosi, Valtiokonttori valitsi järjestelmän suunnittelutyökaluksi koodipohjan osittaisen generoinnin mahdollistavan työkalun.

Rakennetun järjestelmän asiayhteys oli EU:n elpymis- ja palautumistukiväline (RRF)[3], joka lieventää koronapandemian taloudellista ja sosiaalista vaikutusta sekä tekee EU:n talouksista kestävämpiä ja resilienssejä sekä edistää vihreää siirtymää ja digitalisaatiota. Jokainen EU:n jäsenvaltio on laatinut kansallisen elpymis- ja palautumissuunnitelman (Recovery and Resilience Plan, RRP), jota rahoittaa Euroopan komissio. RRP-järjestelmä[4] on tarkoitettu keräämään tietoja Suomen elpymis- ja palautumissuunnitelma[5] toimeenpanosta ja raportoimaan tietoja Suomesta Euroopan komissiolle.

[1] Lano, K., & Siala, H. (2024). Using model-driven engineering to automate software language translation. Automated Software Engineering, 31(1), 20. https://link.springer.com/article/10.1007/s10515-024-00419-y

[2] David, I., Latifaj, M., Pietron, J., Zhang, W., Ciccozzi, F., Malavolta, I., Raschke, A., Steghöfe, J.-P., & Hebig, R. (2023). Blended modeling in commercial and open-source model-driven software engineering tools: A systematic study. Software and Systems Modeling, 22(1), 415–447. https://link.springer.com/article/10.1007/S10270-022-01010-3

[3] https://commission.europa.eu/business-economy-euro/economic-recovery/recovery-and-resilience-facility_en

[4] https://www.valtiokonttori.fi/palvelut/muut-palvelut/suomen-elpymis-ja-palautumissuunnitelman-toimeenpano-ja-seuranta/#tietojarjestelmat

[5] https://commission.europa.eu/business-economy-euro/economic-recovery/recovery-and-resilience-facility/finlands-recovery-and-resilience-plan_en

2 Mallipohjainen kehittäminen ketterässä viitekehyksessä

Ketterät kehittämistavat ovat lisänneet kehittäjien autonomisuutta. Vastapainona liiketoimintaosaajille tarkoitetut koodiminimoidut työkalut ovat pyrkineet edesauttamaan soveltuvuusselvitysten tekemistä ilman koodaustaitoja. Tässä luvussa tarkastellaan mallipohjaista kehittämistä pääasiassa RRP-järjestelmän kehitystyön (Kuva 1) yhteydessä tehtyjen havaintojen ja käytänteiden perusteella ja käytetään RRP-järjestelmän kehittämistä esimerkkinä. Mallipohjaisesti ja ketterästi voidaan kehittää myös monilla eri tavoilla.

Kuva 1. RRP-järjestelmän toisiin tukevat kehitysvaiheet

Kun tietojärjestelmää kehitetään sekä ketterästi että mallipohjaisesti, on tärkeää tunnistaa molempien viitekehysten asettamat reunaehdot ja hyödyntää niitä toisiaan tukien. Mallipohjaisuudessa on kyse siitä, että käsitteet muutetaan malliksi, jonka pohjalta luodaan tietojärjestelmä automaattisesti. Tekemisen keskiössä on siis tietomalli ja sen mallintaminen. Tietojärjestelmän tavoitteet, vaatimukset ja muutokset hahmotetaan tietomallin muutoksina. Pitää kuitenkin huomata, että mallipohjainen kehittäminen ei tarkoita mallilähtöistä kehittämistä eikä reunaehtojen tunnistaminen tarkoita mallintamisen ehdoilla kehittämistä. Suurin osa koodista tuotetaan generoimalla, ja tyypillisesti generointi tehdään työkalulla, jonka käyttäminen ei vaadi koodaustaitoja, vaan mallintaminen vaatii tietomallin hyvää ymmärrystä ja riippuvuuksien hahmottamiskykyä.

Ketterässä kehittämisessä keskiö on käyttäjässä ja käyttäjälle tuotettavan lisäarvon maksimoinnissa. Tämän on oltava kaiken ydin ja selkeä lähtökohta myös silloin, jos tietojärjestelmää kehitetään mallipohjaisesti. Voidaan ajatella, kun toteutuksesta automatisoidaan yli puolet, tietojärjestelmään valmistuu uusia ominaisuuksia mahdollisesti nopeammin kuin perinteisemmällä ohjelmistokehittämisellä, mutta nopeus ei välttämättä tuota käyttäjälle lisäarvoa, jos ei tehdä oikeita asioita.

RRP-järjestelmän kehittämisessä oli käytössä kahden viikon sprintit. Mallipohjainen ketterä kehityssprintti alkaa luonnollisesti sillä, että malliin tehdään tämän sprintin sisällön edellyttämät muutokset mallintajalla ja uusi malli generoidaan. Ideaalitilanteessa generoidussa mallissa on mukana myös mahdollisesti aiemmin tehty räätälöity toteutus, mutta mikäli ei ole, generoitu malli yhdistetään olemassa olevaan toteutukseen ja tuloksen päälle lisätään tarvittavat räätälöinnit. Syklittäisen tekemisen onnistuminen edellyttää hyvää sprinttisuunnittelua: sprintin sisältö on rajattu hyvin etukäteen ja jokainen sprinttiin tuleva kohde (item) on määritelty ja jalostettu (refine) riittävällä tarkkuudella, jotta tunnistetaan sen vaatimat mallimuutokset sekä vaikutukset olemassa olevaan toteutukseen ja mahdolliset muut riippuvuudet. Kun sprintin suunnitteluun on käytetty riittävästi aikaa koko scrum-tiimiltä, pienennetään riskiä sille, ettei generoidussa mallissa olisi huomioitu kaikkia sprintin aikana tarvittavia muutoksia.

Sekä mallintajassa tehdyt muutokset että sprintin aikana tehdyt räätälöinnit on luonnollisesti testattava. Sprintin päätteeksi tuotokset paketoidaan ja viedään hyväksymistestauksen kautta tuotantoon. Hyväksymistestauksen laajuus on hyvä rajata huolellisesti etukäteen kuitenkaan estämättä tutkivaa testaamista. Regressiotestaus on hyvä automatisoida mahdollisimman laajasti myös mallipohjaisessa kehittämisessä.

Yhtä tärkeää kuin paketointi ja hyväksyntätestaus sprintin päätteeksi on pitää kehitystiimin kesken retrospektiivi, jossa fasilitoidusti keskustellaan siitä, mikä on sprintin aikana mennyt hyvin, mitä voisi vielä parantaa ja mitä lähdetään parantamaan ja miten. Tässä voidaan myös hioa mallipohjaista toimintatapaa, jos tunnistetaan, ettei sen kaikkia hyötyä ole otettu käyttöön tai että se hidastaa tai vaikeuttaa tiimin työtä.

Jo ennen kehityssprinttiä on käyty tiivistä vuoropuhelua käyttäjien ja muiden sidosryhmien kanssa, jotta kirkastetaan tavoitteet ja tunnistetaan tärkeimmät kehityskohteet. RRP-tietojärjestelmän kehityksessä on korostettu ihmiskeskeistä kehittämistä, jossa kehittäminen ei edennyt teknologia edellä, siitäkään huolimatta, että projektissa on kokeilussa uusi kehittämisen tapa ja uusi mallinnustyökalu. Tärkeintä on tunnistaa käyttäjäprofiilit, käyttötapaukset ja näiden eri tarpeet ja pyytää palautetta käyttäjiltä säännöllisesti, sillä muuten lisäarvon tuottaminen jää lopulta arvailuksi.

Kuitenkin on huomioitava, että valitut työkalut vaikuttavat merkittävästi mallipohjaiseen kehittämiseen. Esimerkiksi LeBlancin Designer edellyttää Azure-teknologia-arkkitehtuuripohjaisuutta. Lisäksi suurin osa koodipohjasta generoidaan ja käyttöliittymälomakkeet pohjautuvat entiteettimäärittelyihin. Mallipohjainen kehitys ei poista tarvetta määrittely- ja dokumentointityölle ─ uudet ominaisuudet, muutokset ja niiden vaikutukset on määriteltävä hyvin, jotta varmasti tunnistetaan tarvittavat mallimuutokset ja mahdolliset räätälöinnit sekä osataan pilkkoa nämä hyvissä ajoin ennen sprintin käynnistämistä. Dokumentit pidetään ajan tasalla ja sopivan laajoina, jotta ratkaisu pysyy ylläpidettävänä eikä henkilösidonnaisuutta pääse syntymään, muttei kuitenkaan niin laajana, että dokumentoinnin päivitystyö on huomattava ponnistus jokaisella sprintillä.

DevSecOps tarkoittaa sitä, että sama tiimi kehittää (Dev) sekä vastaa tietoturvasta (Sec) ja julkaisemisesta ja tuotantokäytöstä (Ops). Kehittäjät suunnittelevat, koodaavat, rakentavat ja testaavat ohjelmistosovellusta (Dev), varmistavat, että koodissa ei ole haavoittuvuuksia (Sec), ja julkaisevat, valvovat tuotantokäyttöä ja korjaavat mahdollisia ongelmia (Ops). DevSecOps-mallissa huolehditaan siitä, että ratkaisu säilyy uusien ominaisuuksien kehittämisen aikana myös ylläpidettävänä ja tietoturvallisena. Vanhenevat komponentit päivitetään ja mahdolliset haavoittuvuudet ja tietoturvapoikkeamat havaitaan ja korjataan. Ratkaisu skannataan[6] säännöllisesti vaatimustenmukaisuuden, tietoturvallisuuden ja laatukriteerien varmistamiseksi sekä riippuvuuksien havaitsemiseksi, ja havainnot, vaadittavat korjaukset ja muut toimenpiteet päätyvät tarvittaessa backlogille. Tyypillisesti havainnot eivät ole kriittisiä eivätkä vaadi välitöntä reagointia, mutta jotta nämä päätyvät kehityksen backlogille, on niiden kuljettava tuoteomistajan kautta. Tyypillisesti parhaat tulokset saadaan tiimin kanssa keskustellen eikä yksisuuntaisesti raportoiden.

Mikäli mallin muutokset tehdään helppokäyttöisellä työkalulla, jonka käyttö ei vaadi koodaustaitoja, voidaan muutosten tekeminen sisällyttää esimerkiksi tuoteomistajan tehtäviin. Tuoteomistaja tuntee tietomallin ja määrittelyt sekä vastaa niistä, jolloin hänen pitäisi ymmärtää tietojen väliset suhteet ja esimerkiksi mahdolliset käsittelysäännöt ja roolipohjaiset käyttöoikeudet. Tällöin sujuvinta on, jos tuoteomistaja tekee muutokset itse.

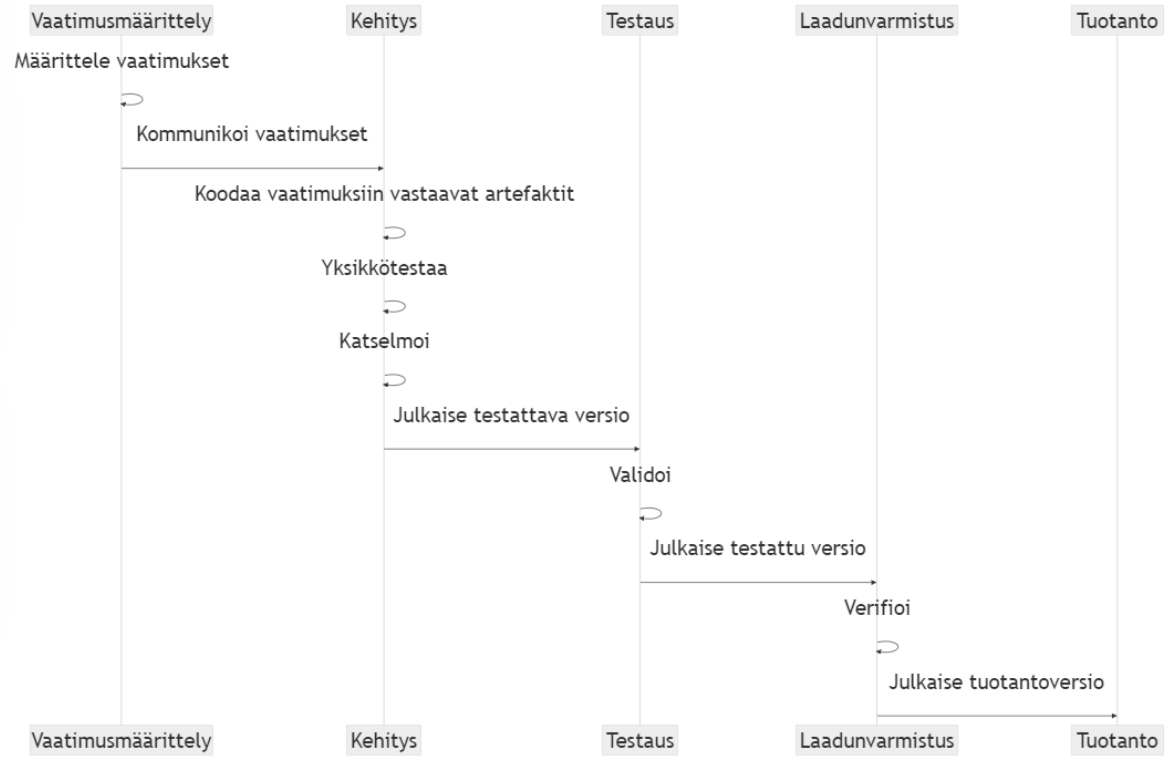
Osa mallintamista on myös käsittelysääntöjen määrittäminen ja asettaminen. Käsittelysäännöillä määritetään esimerkiksi kenttien raja-arvot ja niiden väliset suhteet, joten ne vaikuttavat tiedon oikeellisuuteen ja ohjaavat käyttäjää eli tiedon tuottajaa samalla lailla kuin esimerkiksi lomakkeen kenttien otsikot ja mahdollisesti ohjetekstit. Tästä syystä käsittelysääntöjen pitäisi olla yhtä helposti saatavilla sekä määrittelijälle että loppukäyttäjälle kuin esimerkiksi edellä mainitut käyttöliittymällä loppukäyttäjälle näkyvät tekstit, eikä käsittelysääntöjä siis kannattaisi piilottaa koodiin. Mikäli käsittelysääntöjen luominen, muokkaaminen ja katselu on osa helppokäyttöistä mallinnustyökalua, tällä tavoin voidaan parantaa läpinäkyvyyttä loppukäyttäjille asti.

[6] Skannaustuotteina esimerkiksi Azure Compliance Manager ja Microsoft Defender sekä SonarQube.

3 Tietotuotanto ja tietojärjestelmäkehittäminen

Tietotuotannolla (knowlegde production) tarkoitetaan tässä dokumentissa tietojärjestelmäkehityksessä tarvittavan mallipohjan muodostamista sekä mahdollisesti mallin validointia ja mallin pohjalta tapahtuvaan koodipohjan generointia. Tässä luvussa käytetään sekvenssikaaviota havainnollistamaan erilaisia tietotuotannon viitekehyksiä. Viitekehyksellä (framework) tarkoitetaan toisiinsa liittyviä elementtejä jonkin koostettavan (composable), kuten tietojärjestelmän, hallintaan[7].

Vaatimusmäärittelyt ja niiden perusteella tapahtuva tietojärjestelmäkehitys on vuosikymmeniä edellyttänyt koodauskyvykkyyksiä (Kuva 2). Kehittäjien vastuulla on koodata ja yksikkötestata vaatimuksia vastaavat artefaktit, jotka ovat eri tiedostomuodossa tallennettuja ohjelmistokomponentteja. Ennen kuin ohjelmistokomponenteista muodostetaan testattava kokonaisuus, kehittäjät voivat katselmoida toistensa tuotokset. Testauksessa validoidaan eli varmistetaan, että ohjelmistokokonaisuus on vaatimusmäärittelyjen mukainen eli että on toteutettu oikeita asioita. Laadunvarmistusvaiheessa testaukseen otetaan usein mukaan ohjelmistoa käyttäviä ihmisiä, jotta voidaan verifioida eli varmistaa, että ohjelmistokokonaisuus toimii oikein. Tuotantoonsiirtopäätös on tyypillisesti tuoteomistajan tekemä version julkistamispäätös, jonka yhteydessä informoidaan version ominaisuuksista ja tarvittavista käyttöönottotoimista ja päivitetään dokumentaatio.

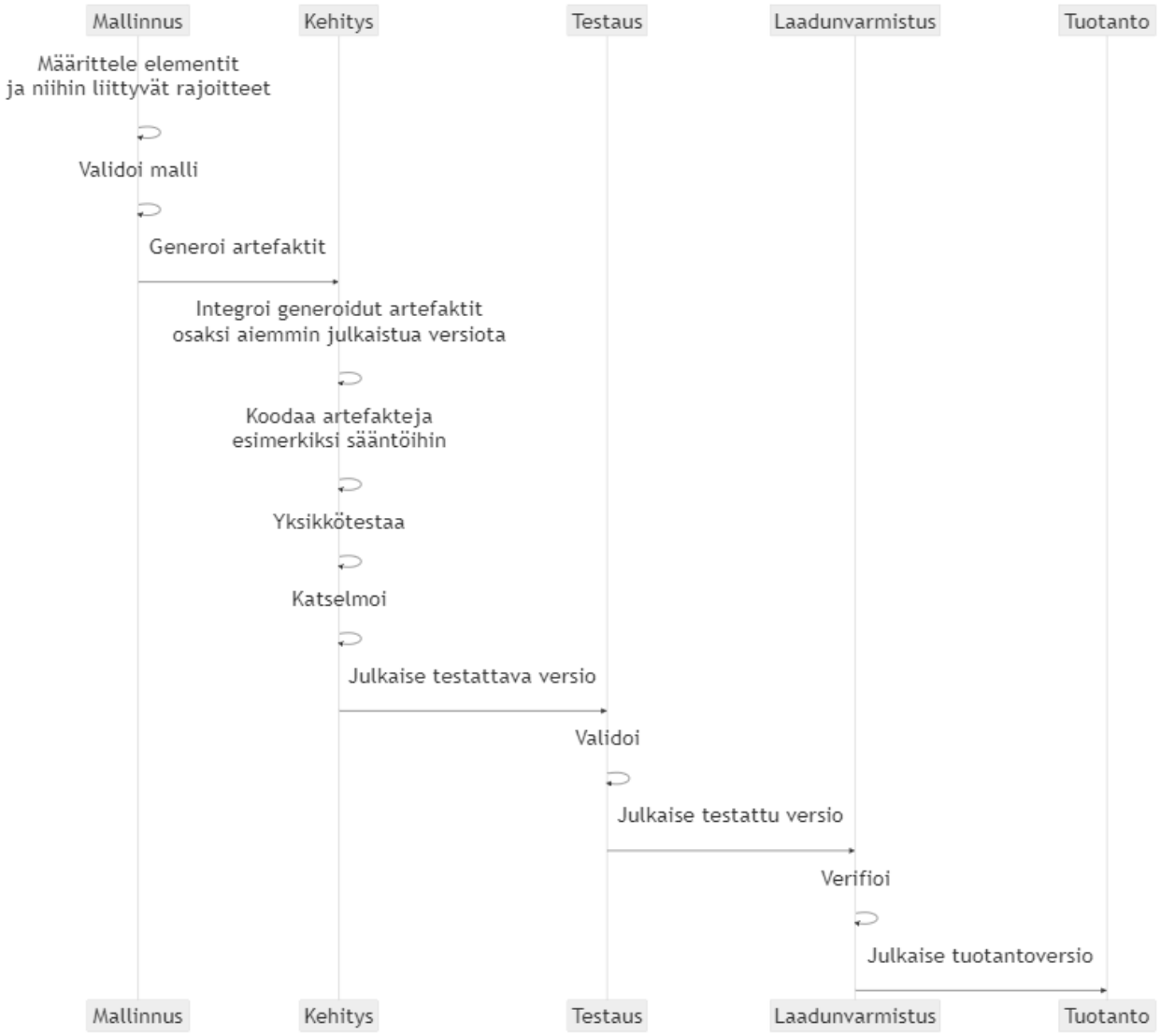


Kuva 2. Koodauskeskeinen tietojärjestelmäkehitystapa

Validointi liittyy syntaksiin eli kielen lauseopin tarkistamiseen. Esimerkiksi tietojärjestelmäkehityksessä validoinnissa tarkastetaan entiteettien välinen viite-eheys ja kenttien tietotyyppiperustaiset rajoitteet, jos niitä on mahdollista määritellä. Verifiointi liittyy semantiikkaan eli kielen merkitysopin tarkastamiseen. Esimerkiksi tietojärjestelmäkehityksessä verifiointi on usein erilaisten syötetietojen vaikutuksien arviointia näyttökenttien esiintymisiin ja syöttökenttien arvoihin.

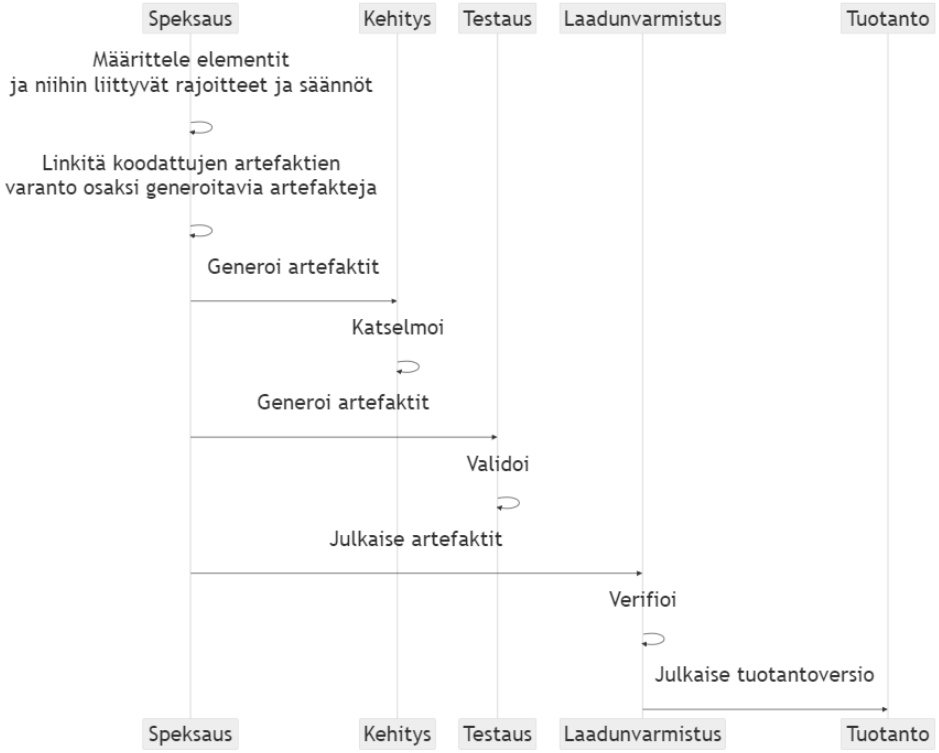
[7] ISO 56008:2024(en) Innovation management — Tools and methods for innovation operation measurements — Guidance; ISO 24143:2022(en) Information and documentation — Information Governance — Concept and principles; Unified Compliance Framework, term 572

Seuraavaksi tarkastellaan tietojärjestelmäkehityksen painopisteen siirtymistä koodauspainotteisesta kehitysvaiheesta mallinnus- tai speksausvaiheisiin. Vaatimusmäärittelyvaihe on sekvenssikaaviotasolla integroitunut osaksi mallinnus- ja speksausvaiheita. Esimerkkinä validoitavasta mallista ja mallipohjaisen kehittämisen tietojärjestelmäkehityksestä on RRP-järjestelmä, jossa LeBlanc Designeriä käytetään suunnitteluprosessissa (Kuva 3) nimeltään mallinnus, ja lopputuloksena on datamodel.json-tiedosto. JSON-tiedostosta generoidaan koodipohja, joka integroidaan olemassa oleviin ohjelmistokomponentteihin. Kehitysvaihe muuttuu sen osalta, ettei kehittäjien tarvitse aloittaa koodaustyötä tyhjästä, vaan täydentää manuaalista koodausta vaativia asioita, kuten käsittelysääntöjä, koodipohjaan kuuluviin tiedostoihin.



Kuva 3. Esimerkki mallipohjaisesta tietojärjestelmäkehitystavasta

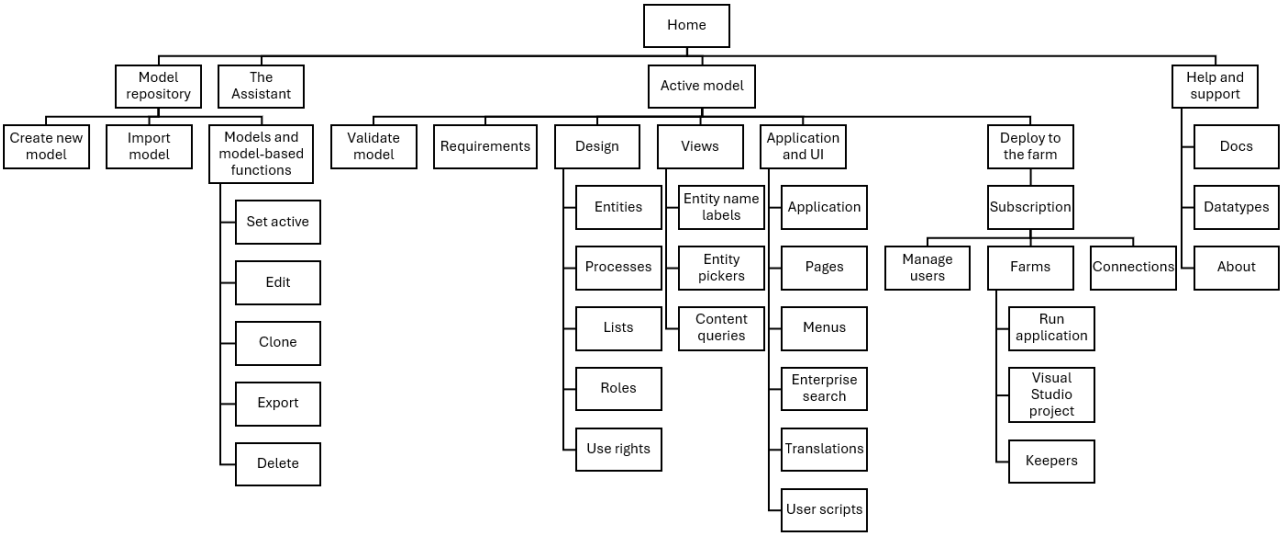
RRP-järjestelmäkehityksessä käytettiin LeBlanc Designerin aiempaa versiota, josta puuttui joitakin nykyisistä ominaisuuksista, minkä vuoksi tehtiin toivottua enemmän räätälöityä kehittämistä. LeBlanc Designerin uusimmassa versiossa (6.9.0) mallipohja voidaan muodostaa AI-avusteisesti eli kehotteiden (prompts) avulla tehdään malli johonkin käyttötarkoitukseen. Kun malli on muodostettu ja validoitu, niin tietojärjestelmä julkaistaan IaaS-palveluna ostettuihin Azure-resursseihin (Kuva 4). Kehitys- ja testausvaiheissa generoidaan tarvittavat artefaktit sekä frontendiin että backendiin. Kun tietojärjestelmä on valmis julkistukseen, julkistus voidaan tehdä asiakkaan osoittamaan laadunvarmistusympäristöön (Quality Assurance), jossa varmistetaan automaattisella ja/tai manuaalisella testauksella, että tietojärjestelmä täyttää määritellyt vaatimukset ja toimii suunnitellulla tavalla. Hyväksytyn laadunvarmistustestauksen jälkeen tietojärjestelmä on valmis siirrettäväksi tuotantoympäristöön. Tässä yhteydessä myös tarkastetaan, että dokumentaatio on ajan tasalla.



Kuva 4. Esimerkki semiformaalista tietojärjestelmäkehitystavasta

4 Mallinnuselementit

RRP-järjestelmäkehityksessä käytössä ollut LeBlanc Designer -suunnittelutyökalu[8] (Kuva 5) mahdollistaa JSON-tiedostomuotoisen tietojärjestelmän mallin tekemisen ja sen validoinnin.

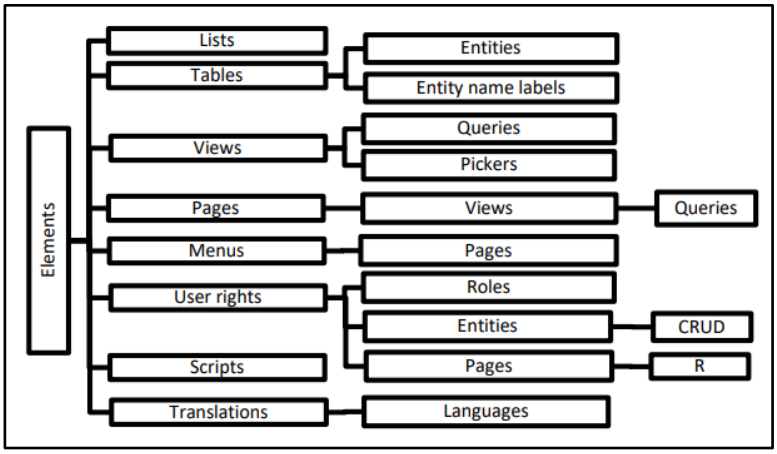


Kuva 5. LeBlanc Designerin toiminnallisuudet 1.6.2024

Seuraavilla elementeillä rakennetaan malli ja tehdään sen muutokset (Kuva 6):

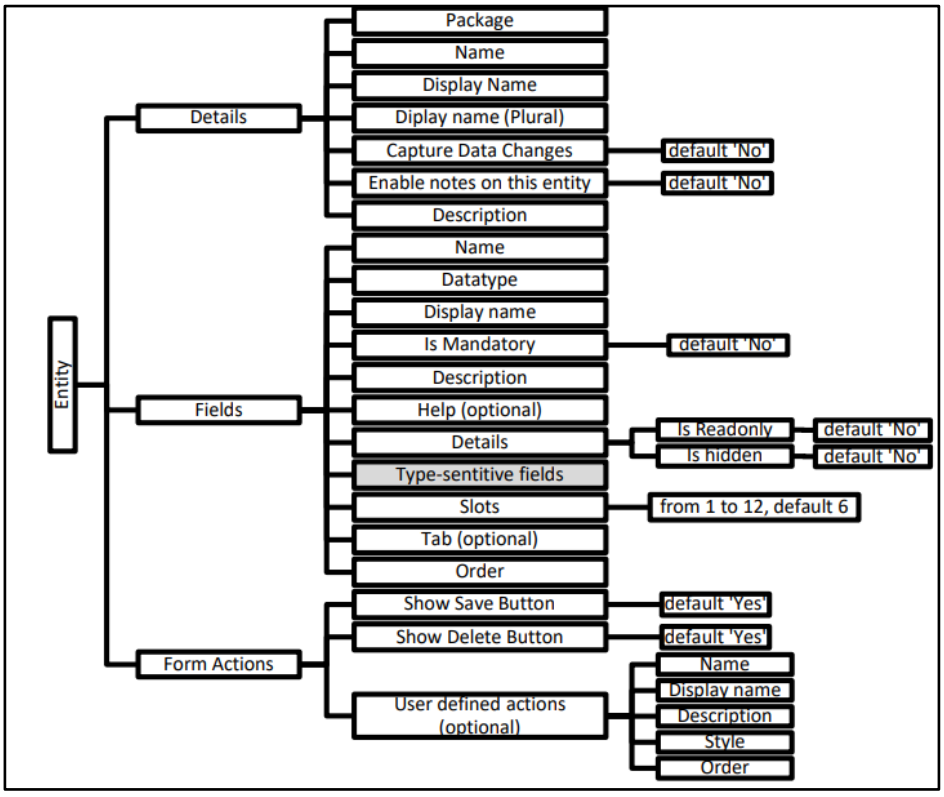
- Listat (lists) ovat luetelmia ja niitä käytetään, kun kenttätyyppi on lista.
- Taulukot ovat entiteettejä (entities), joilla on nimiöintitiedot (entity labels) näkyvissä sivuilla, esimerkiksi, kun luetellaan entiteetti-instansseja.
- Näkymät (views) ovat joko kyselyitä tai valitsimia.
- Kyselyt (queries) ovat nimettyjä SQL-kyselyitä, jotka noutavat tietoja tietokannasta taulukoituihin elementteihin.
- Valitsimia (pickers) käytetään, kun tietty entiteetti osoitetaan linkkikentällä, jonka konkreettinen merkitys on määritellä valitut sarakkeet ja niiden sisältö ja antaa käyttäjän valita yksi listalta.
- Kyselyt voidaan liittää sivuihin. Jos sivu (page) on entiteettilomake, se generoidaan; muuten se koodataan.
- Valikot (menus) ovat hierarkkisia ja niissä on linkkejä sivuille.
- Käyttöoikeudet (use rights) ja niiden määrittely rooleille (roles) määrittelevät, mitkä roolit voivat luoda, lukea, päivittää tai poistaa (eli CRUD) entiteettejä tai mitkä roolit voivat nähdä (eli R) mitkä sivut.
- Skriptit (scripts) ovat C#-kielisiä komentosarjoja erilaisille sovellustapahtumille kuten erilaisille käsittelysäännöille.
- Jokaisella luonnollisella kielellä (languages), kuten English tai Finnish, on numero ja koodi. Kaikki käännettävät tekstielementit (esim. kuvaukset, nimet ja kyselyjen sarakenimet) voidaan kääntää kohdekielelle käyttämällä konekääntämistä tai manuaalista käännöstä.

[8] https://leblanc.fi/



Kuva 6. Mallipohjaisen kehittämisen elementit (mukaillen LeBlanc Designer)

Entiteettejä (Kuva 7) käytetään määrittämään rakenteellisia (esim. kentät) ja käyttäytymisominaisuuksia (esim. lomaketoiminnot lisäävät toiminnallisuutta lomakkeisiin) sekä hallinnollisia näkökohtia (esim. tietojen muutosten tallennus). Käyttöliittymän suunnittelu on tehty rinnakkain kenttämääritysten kanssa.

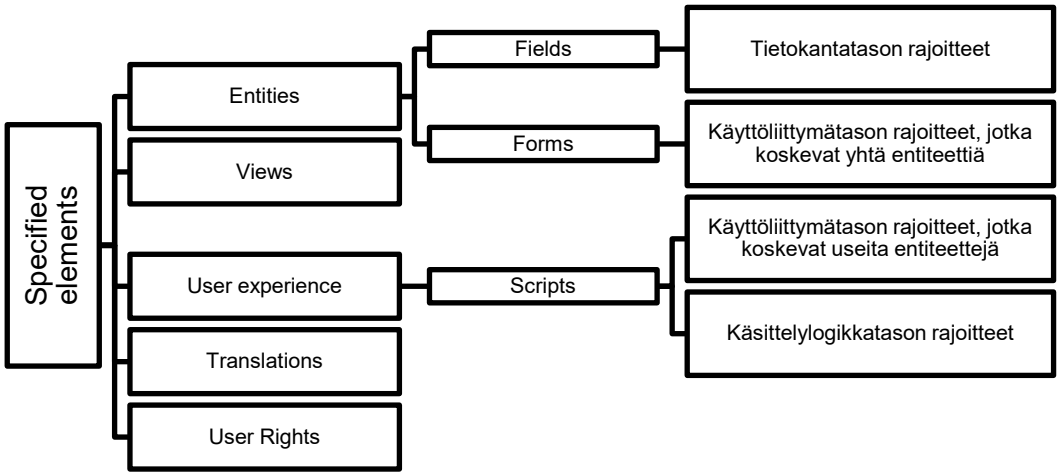


Kuva 7. Entiteettien kenttätyyppikohtaiset määrittelyt (mukaillen LeBlanc Designer)

Suunnitteludatatyypit vaikuttavat luotuihin tietokantatauluihin ja käyttöliittymälomakkeisiin, etenkin jos myös käyttöliittymä generoidaan validoidun mallin pohjalta (ks. liite, Taulukko 1). Yleisemmin voidaan tarkastella ohjelmoitavia asioita suunnittelutyökalun mallinnuselementtien ja komentosarjakielien ilmentyminä (Lano & Siala, 2023).

Yleisesti mallinnuselementtien lisäksi tarvitaan erilaisia rajoitteita (constraints), jotka liittyvät tietokantaan (database), käsittelylogiikkaan (prosessing logic) tai käyttöliittymään (user interface) (Kuva 8). Riippuen käytettävistä suunnittelutyökaluista rajoitteet määritellään ja toteutetaan eri tavoin. Esimerkiksi LeBlanc Designerissä C#-komentosarjoilla (scripts) toteutetaan käsittelylogiikkatason rajoitteet ja useita entiteettejä koskevat rajoitteet (Kuva 9).

Kuva 8. Rajoitteet ja niiden toteutustasot



Kuva 9. LeBlanc Designer ja rajoitteiden toteutusmekanismit

5 Johtopäätökset

Kaikessa kehittämisessä osallistujien tulee puhua yhteistä kieltä ja kommunikoida keskenään sovitulla tavalla. Koska kehitykseen osallistuvilla ihmisillä on erilainen koulutus- ja kokemustausta ja viestimistyyli, kehitysprojektien alussa kannattaa keskustella kehitystyökalujen lisäksi kehittämisessä käytetyistä kielistä ja käsitteistä sekä kommunikoinnissa käytetyistä kuvaustavoista ja niiden tulkintatavoista.

Jotta tietojärjestelmäkehitystä voidaan tehostaa, erilaiset abstraktiot ja kielimuunnokset tulee huomioida eteenpäin suuntautuvassa suunnittelussa (model-driven engineering, MDE) ja käänteis- tai uudelleensuunnittelussa (model-driven reverse or reengineering, MDRE). Valittujen mallinnus- ja toteutusteknologioiden välillä tulee olla johdonmukainen ja selkeä käyttötapa, joka pitää soveltuvuusselvittää ja kommunikoida kehitykseen osallistuville. Erityisesti eritasoisten rajoitteiden suunnittelu- ja toteutustapoihin kannattaa kiinnittää huomiota.

Projektinhallinnallisesti ketterä kehitys tukee mallipohjaista kehittämistä viimeistään siinä vaiheessa, kun kehitysjonoon (backlog) muodostuu tehtäviä, joita allokoidaan ja resursoidaan. Mallinnuksen ja määrittelyjen korostaminen ei tarkoita vesiputousmalliin siirtymistä ─ mallinnuksen ja määrittelyjen korostaminen tarkoittaa, että monimuotoiset artefaktit yhtenäistetään (unify) ja mahdollisuuksien mukaan niistä muodostetaan semiformaaleja (kuten JSON) artefakteja, joista voidaan generoida koodipohja. Tiivis julkaisu- ja palautesykli tukee ketteryydessä tavoiteltua varmistusta käyttäjiltä siitä, että suunta on oikea ja kehityksessä tehdyt oletettamat ovat hyviä.

Valittu toimintatapa toimi RRP-hankkeessa, jossa tietojärjestelmää kehitettiin mallipohjaisesti ja tietojärjestelmä saatiin tuotantokypsäksi noin puolessa vuodessa. Nopean aikataulun onnistumisen takana on useita tekijöitä, eikä ole tietoa siitä, olisiko aikataulu onnistunut, jos tietojärjestelmä olisi rakennettu perinteisesti koodaamalla. Joka tapauksessa tavoitteet saavutettiin, ja tuotantokäytössä on kustannustehokas, helposti ylläpidettävä ja käyttäjien mielestä helppokäyttöinen järjestelmä[9].

Erilaiset pilviympäristöjen hallintamallit ja viranomaisvaatimukset jätettiin tämän katsauksen ulkopuolelle. Esimerkiksi tietosuoja ja -turva, laajemmin vaatimustenmukaisuus, on tietotuotannon perusta. Vaatimustenmukaisuus tarkoittaa julkisessa hallinnossa viranomaisvaatimusten huomioimista eri tavoilla saatujen tietojen manipuloinneissa ja ohjausmekanismeissa, joita hallinnoidaan erilaisten hallintaportaalien avulla. Käytön aikainen järjestelmähallinta tapahtuu yleensä valitussa tenantissa eli pilvipalveluympäristössä. Jos käytön aikana havaitaan tietoturvaskannauksissa tai muissa valvontamekanismeissa poikkeavuuksia, niistä raportoidaan esimerkiksi tietojärjestelmän tuoteomistajalle ja palveluomistajille, jotka vastaavat tietojärjestelmän kehityksestä ja ylläpidosta.

[9] RRP-järjestelmän käytettävyydestä on saatu vain hyvää palautetta järjestelmän käyttäjiltä – sekä palautelomakkeilla että erillisten käytettävyystestausten kautta.

Taulukot ja kuvat

Lähteet

Lähteet on merkitty alaviitteisiin.

Liitteet

Mermaid-syntaksikoodit

Kuva 2
```
sequenceDiagram
    participant V as Vaatimusmäärittely
    participant Dev as Kehitys
    participant Test as Testaus
    participant QA as Laadunvarmistus
    participant Prod as Tuotanto
    V->>V: Määrittele vaatimukset
    V->>Dev: Kommunikoi vaatimukset
    Dev->>Dev: Koodaa vaatimuksiin vastaavat artefaktit
    Dev->>Dev: Yksikkötestaa
    Dev->>Dev: Katselmoi
    Dev->>Test: Julkaise testattava versio
    Test->>Test: Validoi
    Test->>QA: Julkaise testattu versio
    QA->>QA: Verifioi
    QA->>Prod: Julkaise tuotantoversio
```

Kuva 3
```
sequenceDiagram
    participant M as Mallinnus
    participant Dev as Kehitys
    participant Test as Testaus
    participant QA as Laadunvarmistus
    participant Prod as Tuotanto
    M->>M: Määrittele elementit <br> ja niihin liittyvät rajoitteet
    M->>M: Validoi malli
    M->>Dev: Generoi artefaktit
    Dev->>Dev: Integroi generoidut artefaktit <br> osaksi aiemmin julkaistua versiota
```

```
    Dev->>Dev: Koodaa artefakteja <br> esimerkiksi sääntöihin
    Dev->>Dev: Yksikkötestaa
    Dev->>Dev: Katselmoi
    Dev->>Test: Julkaise testattava versio
    Test->>Test: Validoi
    Test->>QA: Julkaise testattu versio
    QA->>QA: Verifioi
    QA->>Prod: Julkaise tuotantoversio
```

Kuva 4

```
sequenceDiagram
    participant S as Speksaus
    participant Dev as Kehitys
    participant Test as Testaus
    participant QA as Laadunvarmistus
    participant Prod as Tuotanto
    S->>S: Määrittele elementit <br> ja niihin liittyvät rajoitteet ja säännöt
    S->>S: Linkitä koodattujen artefaktien <br> varanto osaksi generoitavia artefakteja
    S->>Dev: Generoi artefaktit
    Dev->>Dev: Katselmoi
    S->>Test: Generoi artefaktit
    Test->>Test: Validoi
    S->>QA: Julkaise artefaktit
    QA->>QA: Verifioi
    QA->>Prod: Julkaise tuotantoversio
```

Taulukko 1. Tietotyypit ja niiden vaikutus tietokantaan ja käyttöliittymään (mukaillen LeBlanc Designer)

Suunnittelun tietotyyppi	Pakollinen määrittely	Vapaaehtoinen määrittely	Tietokannan tietotyyppi	Käyttöliittymä-elementit
Attachments (single file)	null	null	ATTACHMENT table as attachment metadata, and STORAGE account to store the actual file	option to upload a single file
Attachments (multiple files)	null	null	ATTACHMENT table as attachment metadata, and STORAGE account to store the actual files	file list with download and upload options
Boolean	null	null	VARCHAR(1) column where T=true, F=False	on-off switch
Content query result	query	button for a new entity creation, either by a new page or by the popup window	null	table with or without the new entity insert button
Date	null	default value; date range (min-max)	DATE column	date picker
Decimal	null	field format instead of default value; numerical value range (min-max), unique database value requirement	DECIMAL(10,2) or specified column	input field
Integer	null	default value; numerical value range (min-max), unique database value requirement	BIGINT column	input field
Link 1:1	entity	null	VARCHAR column containing the target entity id	entity picker
Link 1:N	entity	sorting of the entity instances in ascending or descending order	link table	entity picker
List	list	default value	BIGINT column containing the specified list item number	dropdown list
String	null	default value, maximum field length, unique database value requirement	VARCHAR column	single-line text area
Text	null	default value, maximum field length	VARCHAR column	multiline text area

VALTIOKONTTORIN JULKAISUJA | TIETOHALLINTO 6/2024

Tietotuotantoa tukeva mallipohjainen tietojärjestelmäkehitys

Valtiokonttori

Julkaisija: Valtiokonttori
ISSN/ISBN: 2489-4761 / 978-951-53-3890-7
Sörnäisten rantatie 13, Helsinki | PL 14, 00054 VALTIOKONTTORI
Puh. 0295 50 2000, Faksi 0295 50 3333, www.valtiokonttori.fi

Valtiokonttori
Statskontoret
State Treasury