OULUN AMMATTIKORKEAKOULU

Mikael Leinonen

AKENEON KÄYTTÖLIITTYMÄN YKSILÖIMINEN YRITYKSELLE

AKENEON KÄYTTÖLIITTYMÄN YKSILÖIMINEN YRITYKSELLE

Mikael Leinonen
Opinnäytetyö
Talvi 2022
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tradenomi, Tietojenkäsittelyn tutkinto-ohjelma

Tekijä(t): Mikael Leinonen
Opinnäytetyön nimi: Akeneon käyttöliittymän yksilöiminen yritykselle
Työn ohjaaja(t): Reima Riihimäki
Työn valmistumislukukausi ja -vuosi: Talvi 2022 Sivumäärä: 43

Opinnäytetyössä toteutettiin kehittämistehtävänä tuotetiedon hallinnanjärjestelmä Akeneoon moduuli, jonka avulla sen käyttöliittymään saatiin luotua yksilöity, yrityksen tunnuksia kantava ulkoasu. Toteutuksessa käytettiin opinnäytetyön tilaajan, Pinja Oy:n, toivomia tunnuksia, kuten logoa ja väritystä. Opinnäytetyössä kuvattiin eri vaiheiden toteutusta yhdistämällä Akeneon teoriaa sekä käytännön hyviksi koettuja ratkaisuja Akeneo-kehittäjän suosituksia hyödyntäen. Teorian tueksi haastateltiin kyseistä Akeneo-kehittäjää Akeneon vahvuuksien ja kehittämiskohteiden selvittämiseksi. Opinnäytetyössä kuvataan Akeneon asemaa tuotetiedon hallinnantyökaluna avoimen lähdekoodin markkinoilla. Opinnäytetyössä kuvataan PHP-ohjelmointikehyksen hyödyntämisen välttämättömyydestä ohjelmoinnissa. Akeneo-kehittäminen edellyttää Symfonyn perusteiden osaamista, joten sen toimintaperiaatteita havainnollistetaan. Akeneon modulaarisesta perustasta Symfonyyn kuvataan havainnollistetusti. Opinnäytetyön toteutuksessa käytettiin Dockeria, joten myös tämän työkalun toimintaperiaatteet kuvataan havainnollistetusti. Muut olennaiset paikallisessa kehittämistyössä käytettävät työkalut esitellään yleisesti. Opinnäytetyön jatkokehittämiskohteissa esitellään, miten tehtyä toteutusta voidaan jatkaa esimerkiksi tyylittelyn osalta. Opinnäytetyön keskeisimmät haasteet sekä yleisesti Akeneon kehittämiskohteiksi muodostuneet tekijät käsitellään pohdinnassa.

Asiasanat: Akeneo, PIM, Pinja Oy, paikallinen kehittämistehtävä, Docker

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Option of Business Information Systems

---

Author(s): Mikael Leinonen
Title of thesis: Creating a unique Akeneo user interface appearance for incorporation
Supervisor(s): Reima Riihimäki
Term and year when the thesis was submitted: Winter 2022
Number of pages: 43

Subject of this bachelor's degree thesis was to create a unique appearance for the user interface of the product management tool Akeneo. Pinja incorporated, which ordered the thesis work, needed a unique appearance for the user interface in Akeneo to be implemented further for customers. Implementation of the work was made by combining the theory provided by Akeneo and the practical solutions made by the Pinja Incorporated's developer, who has specialized into Akeneo solutions. Knowledge was gathered in an interview with the developer, where insights and solutions arisen in the development field of Akeneo were documented. Thesis descripes Akeneo´s position in the field of open-source product management tools. Thesis describes the necessity of the use of frameworks in PHP programming with mid-size and large programs. The foundation of the Akeneo relies in the Symfony´s modularity and is therefore described as the backbone of these functionalities. Akeneo utilizes the Docker´s containerization in the development process and is therefore also described. The tools, which were used in making of the module locally, are described in overall. The result of the thesis development is implemented unique user interface. Further development possibilities after thesis regarding, for example, the styling is discussed in the reflection. Reflection also summarizes the challenges during the development process relating to Akeneo.

Keywords: Akeneo, PIM, Pinja incorporated, local development, Docker

SISÄLLYSLUETTELO

1 JOHDANTO

Opinnäytetyön aiheena oli tehdä tuotetiedon hallinnan työkaluun (PIM) Akeneoon moduuli, jolla voidaan muokata sen käyttöliittymää siten, että Akeneo-sovellus voidaan yhdistää Pinja Oy:hyn. Aihe on työelämälähtöinen ja palvelee yrityselämässä esiintyneitä tarpeita luoda yrityksille omannäköinen sekä tunnettavuutta lisäävä ulkoasu, jolla parannetaan yrityksen yhdistettävyyttä Akeneota käyttävänä kumppanina. Käyttöliittymän yksilöiminen lisää yrityksen tunnettuutta ja selkeyttä, jolla vuorostaan parannetaan pidemmällä aikavälillä yrityksen myyntiä. Ohjelmointikehyksenä Akeneo pohjautuu Symfonyyn, joka on modulaarinen ja laajennettava ohjelmointikehys.

Opinnäytetyössä perehdytään Akeneoon tuotetiedon hallinnan järjestelmänä sekä sen nykyiseen asemaan avoimen lähdekoodin markkinoilla. Tietoperustana on viitattu monipuolisiin kirjallisiin tutkimuksiin aiheesta sekä Akeneolla asiakastyössä työskentelevän kehittäjän haastatteluun. Tietoperustassa käydään läpi Symfonya, Dockeria sekä muita nykypäiväisiä paikallisen kehittämisen työkaluja, joilla Akeneo projekti toteutettiin. Opinnäytetyössä kuvataan prosessina Akeneon asentamista paikalliseksi projektiksi, millä on tavoitteena osaltaan lisätä kehittäjille tietoa Akeneon asentamisesta teknisenä dokumentaationa. Teknisen dokumentaation niukkuus on koettu kehittäjien keskuudessa olevan Akeneon osalta kehitettävä kohde, jolla voidaan parantaa kehittäjien Akeneo tietämystä sekä työnlaatua asiakastyössä.

Opinnäytetyössä kuvataan toteutuksen etenemistä paikallisessa kehittämisympäristössä kuvien ja toiminnallisuuksien havainnollistamisella. Kehittämistehtävän toteutusta kuvataan tehtyjen teknisien ratkaisujen pohjalta, jotka perustuvat Akeneon modulaariseen toimintatapaan. Toteutus on yhdistelmä Akeneon omia ohjeita sekä Pinjan Akeneo-kehittäjän näkemyksiä ja suosituksien kokonaisuutta, jotka yhdistetään tässä työssä hyväksi havaittujen osioiden osalta. Jatkokehittämistä tämän opinnäytetyön myötä on kuitenkin edelleen käyttöliittymässä, josta voidaan jatkokehittää yrityksen tarpeisiin soveltuvia osioita, kuten esimerkiksi kategorioiden tyylittelyjä.

2 TUOTETIEDON HALLINNAN TYÖKALUT

2.1 PIM digitaalisten kaupankäynnin järjestelmien ja tuotetiedon yhdistäjänä

PIM tulee englanninkielisestä lyhenteestä Product Information Management Systems. Suomeksi käännettynä tämä tarkoittaa tuotetiedon hallinnan työkalua. PIM-järjestelmä on kehitetty ensisijaisesti yhtenäistämään useamman lähteen tuottamaa lähdetietoa keskitetyksi kokoelmaksi, jota voidaan käyttää eri verkkokaupan työkaluissa. PIM tehostaa yrityksien tiedonhallintaa sekä alentaa kustannuksia päällekkäisistä lähteistä aiheutuvasta kuluerästä. Verkkokauppaan tuotavasta tuotetiedosta tulee täten yhtenäisempää sekä riippumatonta esimerkiksi kielellisistä tai alueellisista eroavaisuuksista. (Battistello 2020, 15; Lantz & Agné 2014, 4; Matos 2021, 46).

PIM:n hyödyntäminen kasvavissa organisaatiossa tulee jossain vaiheessa välttämättömäksi, sillä tuotetiedon määrän kasvaessa, sen hallitseminen muuttuu haastavaksi. Nykyaikaisessa liiketoimintamallissa kaupankäynnin keskiössä on nyt ja tulevaisuudessa suurelta osin digitaaliset tuotteet. Esimerkiksi yrityksen kaikilla myyntipisteillä, toimittajilla, tuotantoketjulla sekä markkinoijilla tulee olla yhteneväinen ja ajankohtainen tieto tuotteista, joita verkkokaupassa myydään. Erityisesti tuotteiden sisältämien attribuuttien sekä niiden sidonnaisuuksien ylläpitäminen toisiinsa nähden manuaalisesti voi tulla huomattavan aikaa vieväksi prosessiksi. PIM:n keskittäessä dataa yhteen, tuotteille tehtävät muutokset voidaan muokata parhaimmassa tapauksessa vain yhdessä paikassa. (Roehl-Anderson 2010, 175; Battistello 2020, 14; Abraham 2014, 17).

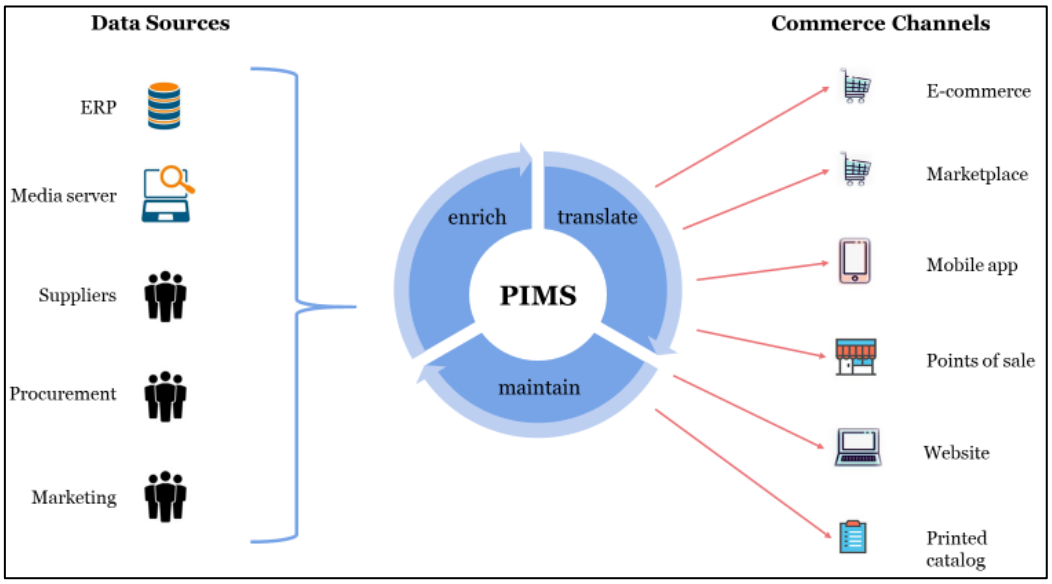
Suurin haaste muodostuu verkkokaupan sisältämistä muuttuvista osista. Suurilla yrityksillä voi olla kymmenittäin tai sadoittain jälleenmyyntipisteitä sekä erilaisia tuotekatalogeja jokaiselle kaupalle. Kaikilla jälleenmyyntipisteillä täytyy yhtä lailla olla myös pääsy tukkuun riippumatta ostajasta tai kaupasta. Yrityksen liiketoiminnan alustoina voi olla käytössä esimerkiksi ERP (toiminnanohjausjärjelmä), PIM:in tuotehallinta sekä CRM (asiakkuudenhallintajärjestelmä). Verkkokaupasta ostaessa sijainnin, toimituksen noutamisen ja lähettämisen tulisi olla kuluttajan ostokanavasta riippumatonta. Usein käyttäjän ostokokemuksessa tuotteen oston jälkeen sekä tukusta ostosten toimittamisessa jälleenmyyntipisteeseen verkkokaupan toiminnanohjausjärjestelmän (ERP) ylläpitämisessä saattaa esiintyä katkonaisuutta. (Lamont 2016, 10; Abraham 2014, 17–18).

ERP tarkoittaa suomennettuna toiminnanohjausjärjestelmää. ERP:n tehtävänä on yhdistää yrityksen tilaustenhallinta, tuotanto, henkilöstöresurssit, taloudelliset järjestelmät sekä jakelu ulkopuolisille toimittajille ja asiakkaille yhdeksi integroiduksi järjestelmäksi. ERP:iin keskitetyllä tiedolla parannetaan yrityksen inventaarioiden reaaliaikaista tarjonnan ja kysynnän tasapainoa, esimerkiksi vähentämällä ylimääräisiä tukkutilauksia, kun kysyntä on vähäistä. Tuotannon työntekijöiden työ- ja lukumäärää voidaan samaten suhteuttaa markkinoiden tilanteeseen ERP:n käsittelemän datan perusteella. Kuluttajien ostokäyttäytymistä mittaamalla ERP välittää ajantasaisen tilanteen kysynnän ja tarjonnan muutoksista yrityksen toimittajille sekä sen sidosryhmille. (Chen 2001, 374; Abraham 2014, 17).

CRM:ää, eli asiakkuudenhallintajärjestelmää, käytetään nykyään kuluttajalähtöisenä liiketoiminnan järjestelmänä, jossa dataa louhitaan kiinnostusten kohteiden sekä tulevien trendien ennustamiseksi suuresta kuluttajien joukosta. Datan louhimisessa yhdistyy tilastot, tekoäly sekä koneoppiminen. Tietokoneiden laskentatehon kasvaessa entistä moniulotteisempia ja kattavampia tuloksia voidaan kerätä sekä analysoida. Kerätystä tiedosta luodaan vuorostaan yritysten strategiseen liiketoimintaan työkaluja fokuksen keskittämiseen, riskien hallintaan, kilpailijatilanteeseen sekä muihin markkinoiden muuttujiin nähden. Uudet kuluttajien kulutus- ja vaikuttamismahdollisuudet markkinoilla ovat luoneet vaatimuksia yrityksille seurata yleisönsä ostokäyttäytymistä ja -tottumuksia parhaimman aseman saavuttamiseksi. Kuluttajan luottamuksen saavuttamisella ja sitoutumisella yritykseen saavutetaan tutkimusten mukaan tuottavimmat ja kestävät vuorovaikutussuhteet. Näistä lähtökohdista CRM:n mallia lähdettiin kehittämään kuluttajien tarpeiden kartoittamiseksi. (SAS 2022; Chen 2001, 383).

Kuvassa 1 kuvataan yleisellä tasolla tuotetiedon keskittämistä PIM-järjestelmään, johon tuotetiedot syötetään ja säilötään vain kertaalleen. Tällä tavalla vältetään tiedon useaan kertaan syöttäminen manuaalisesti sekä pirstaloituminen eri järjestelmissä. Tuotteilta, jollaisia ovat muun muassa kulutustavarat, videot, tietopaketit jne., odotetaan nykyisellään liiketoiminnassa korkeaa laatua. Kuluttajan ostaessa korkealaatuisen tuotteen verkkokaupassa hän odottaa saavansa odotuksiaan vastaavan, verkkokaupan tuotetietoon pohjautuvan laadukkaan tuotteen. Tämän takia tuotetieto on standardisoitava, eli sen sisällöstä muodostetaan yhtenäinen kokonaisuus, jonka laatu säilyy korkeana sen kulkiessa tuoteketjussa ensimmäisestä tuoterikastuksesta alkaen. Tuoteketjun ensimmäinen vaihe on tuotteen perustaminen, jossa tuotteelle asetetaan yleiset ominaisuudet, kuten paino, korkeus, leveys, syvyys, hinta jne. Prosessin ensimmäisestä vaiheesta käytetään nimitystä

tuotetiedon rikastaminen. Tuoteketjun rikastamisen viimeisessä vaiheessa on tuotetiedolla rikastetun tuotteen toimittaminen kuluttajalle hänen valitsemallaan tavalla. (Grizaut 2018, 63; Battistello 2020, 4, 15; Abraham 2014, 18).



Kuva 1. Pimin datalähteet sekä lähtevät kanavat mukaillen Grizauta 2018.

Kuluttajaa pyydetään ostoprosessin jälkeen jakamaan ostetusta tuotteesta kokemuksiaan, joita vuorostaan hyödynnetään tuotetiedon ylläpitämisen prosessissa. Yrityksen eri osastot keräävät kuluttajien ostamien tuotteiden osto- ja käyttökokemuksista tietoa, joilla rikastetaan alkuperäisen tuotteen tuotetietoa paremmin vastaamaan käyttäjien odotuksia ostetulta tuotteelta. Kuluttajien odotusten täyttäminen johtaa täten pidemmällä aikavälillä yrityksen liikevaihdon lisääntymiseen. (Battistello 2020, 50–51).

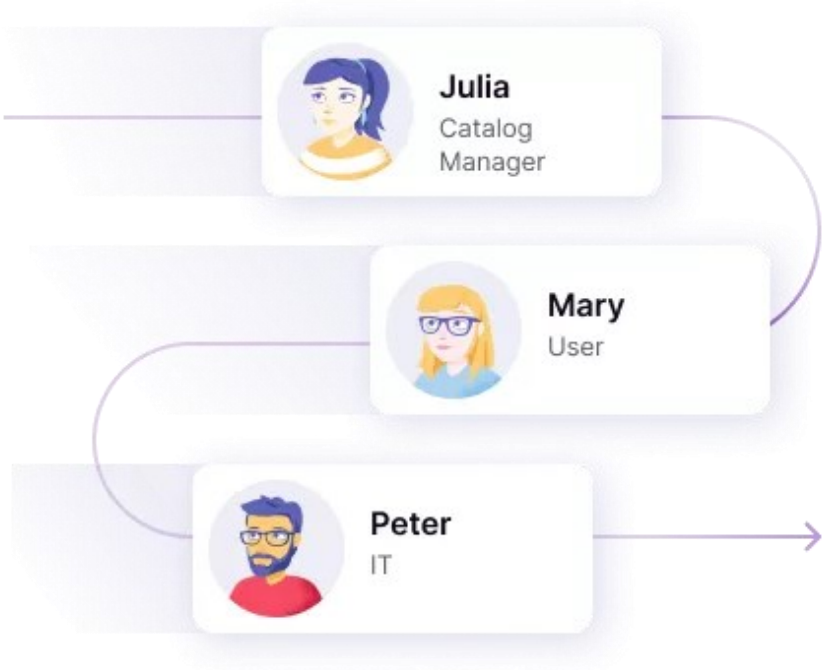
Avoimen lähdekoodin markkinoilla toimivia tuotetiedon hallinnan järjestelmiä ovat esimerkiksi Pimcore, PIMagento ja Akeneo. Jokaisella edellä mainitulla sovelluksella on eri osa-alueilla vahvuutensa. Pimcore tunnetaan yhteensopivuudestaan eri arkkitehtuurien kanssa, tiiviistä integraatiosta verkkokauppa-alustoihin sekä kokemuksesta verkkokauppa-asioinnissa. PIMagento pohjautuu verkkokauppa-alusta Magentolle, jota on laajennettu käsittelemään tuotetiedon rikastamista käyttämällä mm. tietokantaintegraatiota sekä MAM:a (Media Asset Management). Akeneo vuorostaan pohjautuu tehokkaalle modulaarista ohjelmointikehystä hyödyntävälle Symfony 2:lle, joka on monipuolisesti muokattavissa erilaisiin ja erikokoisiin tarpeisiin. Akeneo on ollut eCommerce palkintojen finalisti vuonna 2015. (Eine Bastien, Jurisch & Quint 2017, 9–10; Richter 2022).

2.2 Akeneo

Akeneo nimettiin vuonna 2019 maailmanlaajuiseksi pioneeriksi PXM (Product Experience Management) ratkaisujen tarjoajana tuotetiedon hallinnan työkalujen saralla. Akeneo on arviointien mukaan tarjonnut brändeille ja jälleenmyyjille vastustamattomia käyttäjäkokemuksia monikanavaisessa myynnissä. IDC MarketScapen arvioinnissa oli mukana 14 yritystä, joiden joukosta valittiin viisi johtajaa. Akeneon sijoittumista perusteltiin sen nykyisellä kapasiteetillä sekä strategisella asemalla nopeasti muuttuvassa PXM ympäristössä. Akeneosta arvioinnin antaneet asiakkaat arvioivat sen sijoittuvan keskimääräistä paremmin lähes kaikissa arvioinnin kriteereissä. (Akeneo 2019).

Akeneon sanotaan digitaalisen liiketoiminnan ja yrityssovellusten tutkimusjohtajan mukaan olevan erittäin hyvä valinta keskikokoisille ja sitä suuremmille yrityksille. Valintaa perustellaan korkealaatuisella tuotetiedon säilyvyydellä kaikissa kanavissa, mukaan lukien painetussa mediassa. Akeneota käyttäviä nimekkäitä yrityksiä maailmalta ovat esimerkiksi Sephora, Fossil, Shop.com, Auchan ja Petra Industries. Tutkimusjohtaja Fred de Gombortin mukaan Akeneo soveltuu samalla tavalla niin kuluttajamyyntiin (B2C) kuin yritysmyyntiin (B2B) sen tuotehallinnan helppouden ja automatisoinnin muunneltavuuden perusteella. Tutkimuksessa mukana olleille yrityksille kriteerinä oli Akeneon monikanavaisuus ja eri alustojen välinen yhteensopivuus. Akeneon avulla yrityksen brändit sekä jälleenmyyjät voivat vahvistaa kuluttajien ostokokemusta, lisätä myyntiä, vähentää markkinoille siirtymän aikaa sekä tehostaa tiimien välistä toimintaa. (Akeneo 2019).

Kuvassa 2 kuvataan tuotehallinnan prosessia, jossa rikastettava tieto myynnin ja markkinoinnin hallitsemisessa kulkee yrityksessä eri käyttäjien kautta. Tuotehallinnan työkalut tekevät tuotetiedon hallitsemisen prosessista nopeaa, tehokasta ja tuottavaa. Kaikki tuotteen kanssa tekemisissä olevat työntekijät voivat lisätä oman osa-alueensa asiantuntijuutta tuotteen tietoihin, mikä parantaa tuotteen sisältöä, nopeuttaa pääsemistä julkaisuun sekä mahdollistaa jatkuvan ajan tasalla pysymisen.(Akeneo 2022a).



Kuva 2. Akeneon tuotetiedon rikastamisprosessiin osallistuvat työntekijät sekä kuluttajat (Akeneo 2022a).

Tuotetiedon rikastamisen prosessissa kohtaa kolme eri ulottuvuutta: tekninen data, käyttödata sekä emotionaalinen data. Tekniseen dataan kuuluvat tuotemerkinnät, mitat, värit, ainesosat sekä muut vastaava tuotteen fyysisiin ominaisuuksiin lukeutuvat asiat. Käyttödatassa määritellään missä ja miten tuotteita tulee käyttää. Emotionaalisessa datassa yhdistetään mukaansatempaavia kuvauksia, tuotetarinoita, brändiarvoja sekä yhdistetään olennaisia kuvia, videoita sekä dokumentteja tuotteisiin, jotka helpottavat käyttäjille hankintapäätöksien tekemistä. (Akeneo 2022a).

2.3 Pinja ja Akeneo

Pinja Oy on vuonna 1990 perustettu teknologia-alan yritys, joka toimii usealla eri toimialalla kansainvälisesti. Pinjan toimialoihin lukeutuvat mm. valmistava teollisuus, ympäristön kehittämisen ja kiertotalouden sektori, luonnonvarojen teknologia, terveys- ja hyvinvointialan teknologia, metsätalous sekä digitaalinen verkkokaupankäynti. Opinnäytetyö toteutetaan Pinjan digitaalisen verkkokaupan, Digital Commercen yksikössä. (Pinja 2022).

Pinjalla on valittu Akeneo PIM, sillä siellä työskentelevällä henkilöstöllä on vahva osaaminen jälleenmyynnin markkinoilla tarvittavasta tuotetiedon ylläpidosta, hallittavuudesta sekä ketteryydestä.

Päivittäisessä asiakastyössä työskentelevillä ammattilaisilla on useamman vuoden kokemus Akeneosta. Digitaalisen verkkokaupan yksikössä on luotu laaja-alaisesti eri tasoisia ratkaisuja tuotetiedon työkalujen toteutuksista. Pinja tekee yhteistyötä Akeneon osalta tällä hetkellä noin kymmenen asiakkuuden kanssa. (Akeneo PIM 2022; Tapalinen 2022).

Akeneon suurimmiksi vahvuuksiksi tuotetiedonhallinnan työkaluna on koettu kehittäjien keskuudessa nopeus sekä helppokäyttöisyys. Kehittäjän mukaan järjestelmän monikanavainen tuki sekä yhdistettävyys muihin digitaalisiin kaupankäynnin alustoihin mahdollistaa joustavan työskentelytavan asiakkuudesta riippumatta. Akeneon yksi vahvuus on sovelluksen tarjoama käytettävyys useammalla kielellä. Akeneo on myös muokattavissa laajasti erilaisten ja erikokoisten asiakkaiden tarpeisiin. (Tapalinen 2022).

Suurimmat haasteet Akeneon käytettävyydessä asiakastyössä liittyvät sen hallitsevien tekijöiden määrään. Akeneon kehittämistä hallitsevia kehittäjiä tarvittaisiin kehittämistyössä lisää. Tämän osalta Akeneossa tehtävä kehitystyö on usein muutaman kehittäjän tekemien toteutusten varassa. Ideointi ja vaihtoehtoiset toteutukset useamman osaajan kesken voisi parantaa tuotekehityksen laatua. Tilanne johtuu osittain siitä, että Akeneon kehittämiseksi edellytetään keskimääräistä korkeampaa tietotasoa. Vaatimuksiin lukeutuvat esimerkiksi Symfony ohjelmointikehyksen perusteiden hallintaa sekä laajaa objektiorientoituneen ohjelmoinnin osaamista. (Tapalinen 2022).

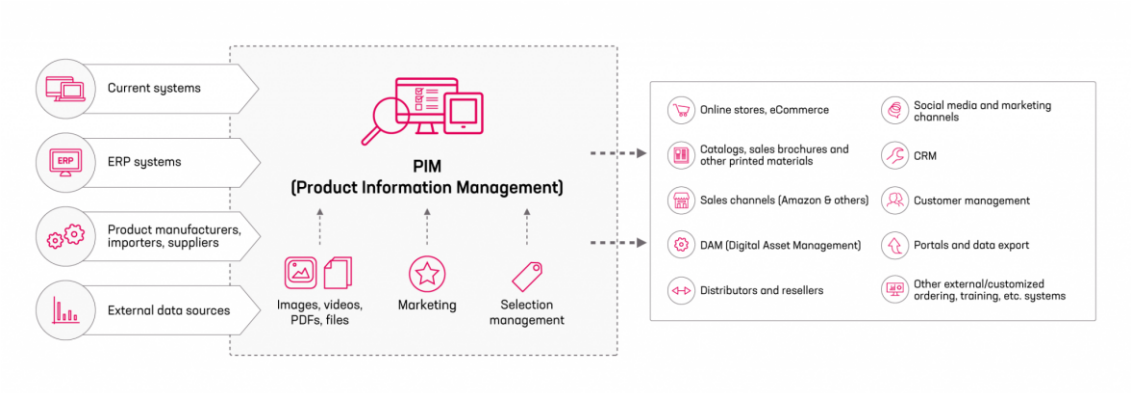
Kehittäjän Akeneoon perehtymisen alkuvaiheessa yksi huomattava haaste on myös kehittämiseen soveltuvan dokumentaation puuttuminen. Kehittäjällä oletetaan olevan perustietämys muista objektiorientoineista sovelluksista sekä niiden monipuolisesta käytettävyydestä Akeneoon siirryttäessä. Akeneon kehittämiseen tarkoitettu dokumentaatio voisikin olla runsaampaa sekä huomioida paremmin lukijoidensa eri lähtötilanteet. Tämä luo myös omat haasteensa asiakkaille tarjottaviin palveluihin. Suurin osa tällä hetkellä saatavilla olevasta dokumentaatiosta on suunnattu Akeneon loppukäyttäjille. (Tapalinen 2022).

Akeneo ja Magento solmivat keskenään Magento Premier Technology Partnership sopimuksen lokakuussa 2019. Sopimuksen keskeisenä asiana Magento antaa tunnustusta Akeneolle avoimen lähdekoodin yrityksenä, joka tarjoaa Magentolle mahdollisuuden hyödyntää korkealaatuista tuotetiedon integroimista verkkokaupan alustalle. Tällä tavalla verkkokaupan asiakkaille pystytään toimittamaan kattavampi sekä yhtenäisempi kokemus eri myynnin kanavissa. (Akeneo 2018; Varanka 2022).

Akeneon ja Magenton yhteistyö mahdollistaa erityisesti Pinjalla toimivien asiakkaiden tarjoamien brändien sekä niiden myyjien tuotedatan laatua ja merkityksellisyyttä kuluttajille. Pinja on Magenton tunnustama verkkokaupan ratkaisuja tarjoava virallinen kumppani (Adobe 2022). Pinjan päätuotteena on Magenton verkkokaupan ratkaisujen toteuttaminen usealle asiakkuudelle. Siten Akeneon ja Magenton tiivis yhteistyö parantaa mahdollisuuksia luoda Pinjalle monipuolisempia palvelukokonaisuuksia verkkokaupan ratkaisuihin (Varanka 2022).

Akeneo on tuotetiedon hallinnan työkaluna avoimen lähdekoodin ratkaisu, joka on perustamisestaan lähtien kasvattanut asemaansa markkinoilla toimijana. Akeneossa voidaan esimerkiksi luoda monipuolisia tuotekatalogeja, joita jaetaan monikanavaisesti myynnissä sekä verkkokaupan alustoilla. Akeneossa yhdistyy erityisesti nopeus, muokattavuus sekä aktiivisen kehittäjäyhteisön tuki. Akeneo on keskitetty PIM ratkaisu, jossa kerätään, hallitaan sekä rikastetaan tuotetietoa monipuolisilla teknisillä ratkaisuilla. (Akeneo 2022b; Matos 2021, 27–28).

Akeneon vahvuuksia on muun muassa se, että se ei pelkästään yhdistä tuotetietoa. Tämän lisäksi mukaan voidaan lisätä markkinointitietoa, kuten käyttäjien yksityiskohtaisia kertomuksia, rikastettuja kuvauksia, hakukoneoptimointia (SEO) sekä muita markkinoinnin työkaluja. (Matos 2021, 10; Akeneo 2018). Kuvassa 3 on havainnollistettu Pinjan käyttämiä sovellutuksia tuotetiedon hallinnan työkalujen käyttökohteita ja mukautuvuutta.



Kuva 3. Pinjan palveluissa käyttämiä tuotetiedon hallintajärjestelmän käyttömahdollisuuksia.

3 OHJELMOINTIYMPÄRISTÖ

3.1 PHP-ohjelmointikehyksen käyttäminen sovelluskehityksessä

PHP on avoimen lähdekoodin ohjelmointikieli, joka on kehittynyt julkaisuvuodestaan 1993 valtavasti. PHP on ohjelmointikielenä verraten helppo opetella myös aloittelijana verrattuna markkinoilla toimiviin konekieliä lähempänä oleviin ohjelmointikieliin, kuten esimerkiksi C tai C++. PHP:n asema erityisesti kaupallisissa web-sovelluksissa on edelleen markkinoiden suurimpia. Esimerkiksi Facebook, Yahoo!, Drupal ja WordPress käyttävät edelleen ohjelmointiratkaisuissaan PHP:tä. PHP:n suurin vahvuus on palvelinpuolen dynaaminen yhdistettävyys ja nopeus selaimessa käytettävän HTML:n kanssa (Hypertext Markup Language). (MacIntyre 2010, 16–17).

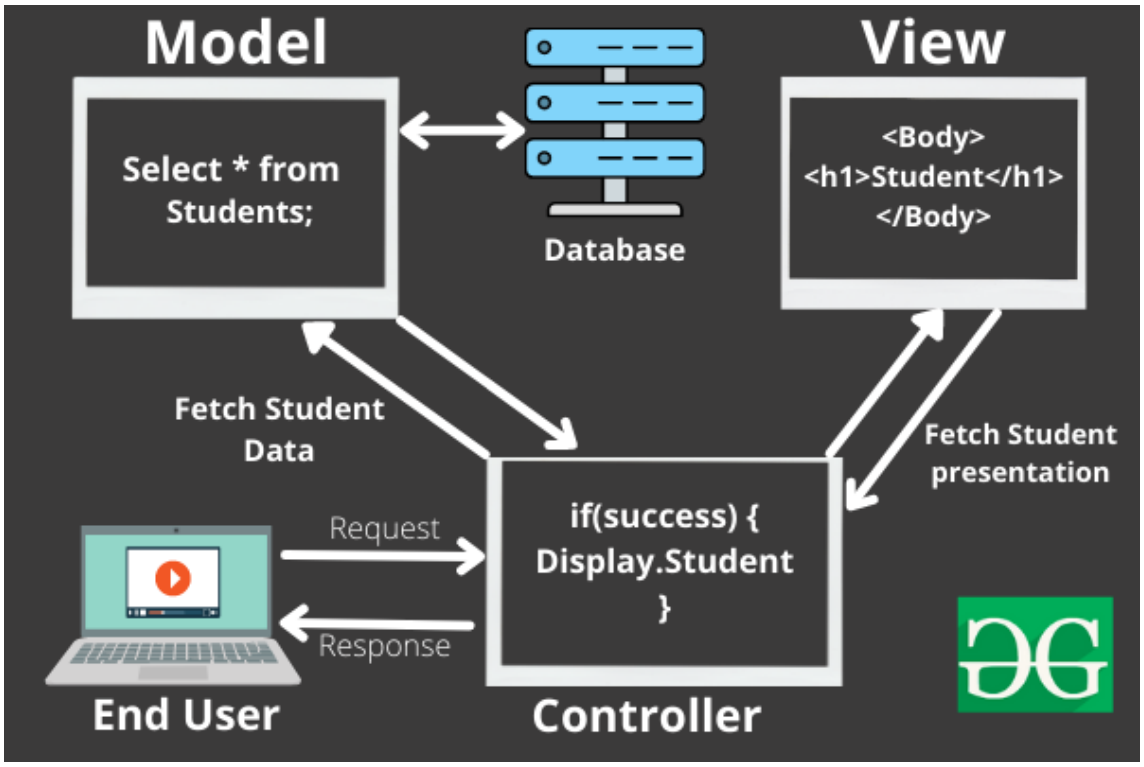
PHP-ohjelmointikehyksen käyttämisellä on sovellusten kehittämisessä merkittävä rooli. Ohjelmointikehysten käytön tarve korostuu suurempien ohjelmistojen luomisessa. Koodin uudelleenkäytettävyyden, testaamisen ja muutosten tekeminen sovelluksessa ilman ohjelmointikehystä aiheuttaa suuremmat riskit erilaisille ohjelman sisäisille virhetilanteille. Ohjelmointikehys ensisijaisesti myös nopeuttaa ja tarjoaa sovellukselle pohjarakenteen, jota voidaan kutsua myös RAD (Rapid Application Development) periaatteen mukaiseksi. RAD periaatteen mukaisesti suunnitellulla sovelluksella on nopea vasteaika, vakaampi käytettävyys sekä vähennetään turhaa toistoa ohjelmakoodissa. (Laaziri, Benmoussa, Khoulji, Larbi & Yamami 2019, 704–705).

Aloittelijoille ohjelmointikehyksen käyttäminen varmistaa lisäksi sovelluksen eri osien yhteensopivuuden toistensa kanssa, vähentää virheitä esimerkiksi päivittämisen yhteydessä sekä parantaa merkittävästi tietoturvaa. Yhteisön merkitys ohjelmoinnissa näkyy sovelluskehyksiä käytettäessä jatkuvana kehittämisenä, kuten löydettyjen bugien korjaamisena ja tiheämpinä päivityksinä. Ajantasainen ohjelmointikehys on myös tietoturvallisempi standardoinnin puolesta. Sovelluskehyksen käyttö vapauttaa aikaa ohjelmoijalle enemmän varsinaiselle sovelluksen kehittämiselle, kun pohjarakenne huolehtii perustoiminnallisuuksien suorittamisesta. (Laaziri ym. 2019, 704–705; Prokofyeva & Boltunova 2017, 51–52).

Sovelluksen kasvaessa, sen eri osien tulee toimia dynaamisena kokonaisuutena. Tähän tarpeeseen on kehitetty vuoden 1988 jälkeen niin kutsuttu MVC (Malli, Näkymä, Käsittelijä) -malli. MVC

yhdistää sovelluksen eri toiminnalliset osat siten, että niitä pystytään muokkaamaan yksittäin sekä ylläpitämään keskitetysti. Selain- (frontend) ja palvelinpuolen (backendin) yhdistämisen merkitys korostuu suuressa sovelluksessa, jossa voi olla käytössä useita eri teknologioita samanaikaisesti. (Pop & Altar 2014, 1173–1175).

Kuvassa 4 on esitetty MVC-mallin yksinkertaistettu kuvaus toiminnallisuudesta, jossa käyttäjä vuorovaikuttaa verkkosivun kanssa. Esimerkissä haetaan läsnä olevien opiskelijoiden tiedot järjestelmästä. Käsittelijä vastaanottaa käyttäjän pyynnön opiskelijoiden listauksesta, joka lähetetään eteenpäin mallille sekä näkymälle. Malli hakee tietokannasta opiskelijoiden tietueen sekä palauttaa löytyneet opiskelijat takaisin käsittelijälle. Näkymä prosessoi tietueen opiskelijoista käyttäjälle helpommin luettavassa HTML muodossa. Käsittelijä palauttaa lopuksi käyttäjälle näkymän luettavassa muodossa, jossa on tietokannasta löytyneet opiskelijat. (MVC Framework Introduction 2022).



Kuva 4. MVC-mallin toiminta käyttäjän lähettämästä vuorovaikutuksesta verkkosivulla. (MVC Framework Introduction 2022).

MVC-mallin mukaisesti rakennetut PHP-sovellukset ovat helposti muokattavissa erillisinä osina, kuten esimerkiksi käyttöliittymän näkymän eristäminen loogisista toiminnallisuuksista. Mallin tehtävä on käsitellä dataan liittyvät toiminnallisuudet, kuten validoinnit, istunnot ja niiden hallinta sekä

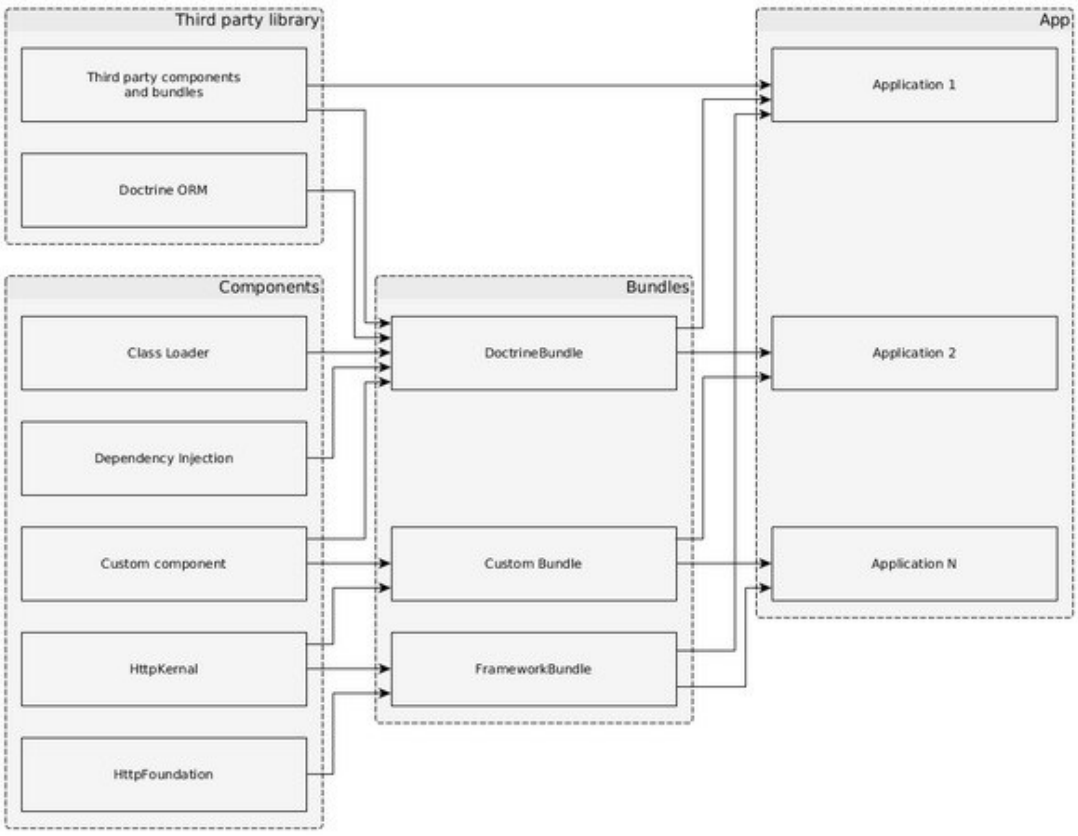
tietokannan rakenne. Mallissa tapahtuu myös sovelluksen yhdistettävyys eri metodien ja luokkien kesken. Näkymän tehtävä on graafisen käyttöliittymän hallinta, jossa ovat esimerkiksi lomakkeet, painikkeet, graafiset elementit jne. Näkymässä sovelluksen eri toiminnallisuuksille voidaan luoda visuaalisia tyylittelyjä sekä kerätään käyttäjän syöttämää dataa yleisimmin HTML, CSS ja JavaScript elementtejä käyttämällä. Käsittelijän tehtävänä on käsitellä kaikkia sovelluksen tapahtumia, kuten vastaanottaa selainpuolelta tulevia pyyntöjä (request) ja yhdistää sitä vastaava vastaus (response) palvelinpuolen kanssa. Käsittelijä yhdistää mallin ja näkymän toiminnat kokonaisuudeksi. (Laaziri ym. 2019, 704–706; Pop & Altar 2014, 1174–1176).

3.2 Symfony ohjelmointikehys

Yksi käytetyimmistä avoimen lähdekoodin PHP-ohjelmointikehyksistä ohjelmoijien keskuudessa on Symfony, jonka ensimmäinen versio julkaistiin vuonna 2005. Yksi syy Symfonyn käytön nopeasti yleistyvälle kasvulle oli tarve luoda edistyneitä ohjelmistoja, joissa on paljon suuren verkkokaupan vaatimia ominaisuuksia. Tällaisia ovat esimerkiksi suuret käyttäjämäärät, vakaus sekä kustannustehokas ylläpidettävyys. Symfonyn suurimpia vahvuuksia oli sen aktiivinen ohjelmoijien yhteisö sekä kattava dokumentaatio, jota kehitettiin aktiivisesti kehitysympäristön kehittyessä. (Zaninotto & Potencier 2007, 4–6).

Symfonyn MVC-arkkitehtuuri noudattaa klassista ohjelmistosuunnittelun rakennetta, joka jakautuu kolmeen eri osioon: Malli, näkymä ja käsittelijä. Symfony rakentaa näiden perusrakenteiden päälle oman lisäkerroksen. Yksi tällainen kerros on selainpuolen ja asetelmien (layout) yhdistäminen yhdeksi sovelluksen tasoksi. Tämä taso säilyy yksittäisenä, vaikka rinnakkaisia käsittelijöitä ja asetelmia olisi useita. Selainpuolen käsittelijät, tietokannan mallit sekä luokat generoidaan automaattisesti Symfonyn toimesta käyttämällä komentokehotteita.(Zaninotto & Potencier 2007, 20–22).

Symfony käyttää rakenteessaan, toisin sanoen teknisissä ratkaisuissaan, monia erilaisia teknologioita. Uusimmissa ohjelmistoissa pienin Symfonyn kehittämiseen käytettävä PHP:n versio on 5. PHP 5 hyödyntää aiempia versioita keskitetymmin objektiorientoituneen ohjelmoinnin käyttämää paradigmaa (oppirakennelmaa), kuten luokkia, objekteja, metodeja sekä superluokilta perimistä. (Zaninotto & Potencier 2007, 6–8). Kuvassa 5 kuvataan tarkemmin Symfonyn relaatiorakennetta, jossa yhdistetään useammat komponentit toimimaan ohjelmointikehyksessä kokonaisuutena.



Kuva 5. Symfonyn rakenteen relaatiomallinnus, jossa ohjelmiston eri osat muodostavat ohjelmointikehyksen kokonaisuuden. (Symfony - Architecture 2022).

Symfonyssa ORM:a (Object-Relational Mapping) käytetään tietokantojen hyödyntämiseen Doctrine-kirjastoa, joka toimii olio-relaatiokartoittajana. Tietokannat ovat relationaalisia, kun taas PHP ja Symfony ovat vuorostaan objektiorientoituneita. Doctrine hyödyntää tietokantojen sisältöä kääntämällä niiden relationaalista logiikkaa objektiorientoituneeksi logiikaksi. Doctrine koostuukin objekteista, jotka mahdollistavat pääsyn tietokannan dataan muuttamalla SQL kutsut tietokantaan objektien mallinnukseksi. Tämä mahdollistaa esimerkiksi tietokannan tyypin, kuten SQLiten tai MySQL:n, muuttamisen saumattomasti kesken projektia toiseen tietokantaan (Zaninotto & Potencier 2007, 6–8).

Symfonyn projektin rakenne koostuu useasta erillisestä palvelusta ja operaatiosta, jotka on koottu yksittäisen domainin alaisuuteen, ja jotka jakavat saman objektin mallinnuksen. Projektissa operaatiot on luokiteltu itsenäisiksi sovelluksiksi, jotka voivat suorittaa toimintojaan itsenäisesti mutta jakavat yhteisen tietokannan. Tällaisia ovat esimerkiksi selain- ja palvelinsovellus. Yleisten selain- ja palvelinsovellusten lisäksi Symfonyssa voi olla lukemattomia pienikokoisia itsenäisiä sovelluksia

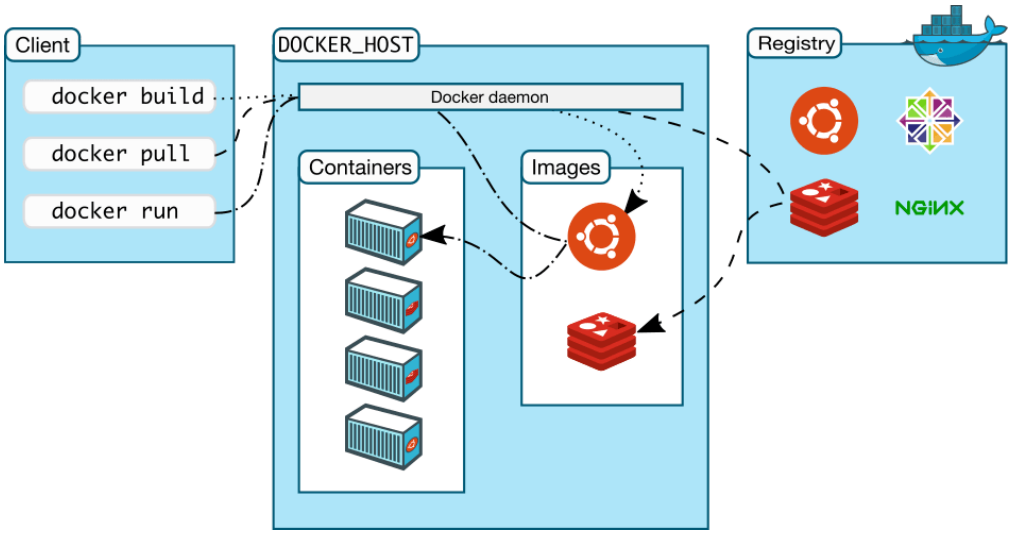
samanaikaisesti erinäisiin toimintoihin. Näitä yksittäisiä sovelluksia kutsutaan moduuleiksi. Esimerkkinä ostoskori voi olla moduuli, josta löytyy muun muassa seuraavanlaisia toiminnallisuuksia: lisää ostoskoriin, näytä ostoskorin sisältö, päivitä ostoskori jne. (Zaninotto & Potencier 2007, 24–26).

3.3 Docker

Ohjelmistokehityksessä tärkeässä asemassa nykyisessä ja tulevaisuuden ohjelmoinnissa on yhdistettävyys, sillä ohjelmistojen kehittäminen on siirtymässä käyttämään pilvitallentamista ensisijaisena tallentamisen muotona. Riippumattomuus eri alustojen keskinäisistä eroista tai ohjelmoijien paikallisen ohjelmointiympäristön kokoonpanosta on välttämättömyys keskisuurissa ja suurissa projekteissa. Docker on vuonna 2013 luotu avoimen lähdekoodin ohjelmisto, jossa pilvitallentaminen on keskeisessä roolissa. Docker yksinkertaistaa eri kehitystyön versioiden hallintaa, sillä sen pilvipohjaisessa ratkaisussa tiedostot voidaan jakaa eri kokoisten työryhmien kesken saumattomasti ja nopeasti. (Nickoloff & Kuenzli 2019, 28–30; Turnbull 2014, 33).

Docker on kehittänyt oman teknologiansa versioiden hallintaan, joita kutsutaan Docker-konteiksi (container). Docker-konttien hyödyntäminen ohjelmistokehityksessä muun muassa pienentää viivettä, kun kaikenkokoiset työryhmät voivat saumattomasti julkaista kehitystyönsä samassa ohjelmointiympäristössä kontteja käyttämällä. Tämä työskentelytapa kuluttaa vähemmän resursseja sekä ylläpitää tehokkaasti tehtävää kehitystyötä myös tuotannossa. Yhteensopivuutta tai skaalautuvuutta ei tarvitse siten erikseen konfiguroida eri kehittäjien ympäristöihin sidotusti, sillä keskitetty konttien käyttö säilyttää yhteensopivuuden kaikkien kesken. (Nickoloff & Kuenzli 2019, 28–31).

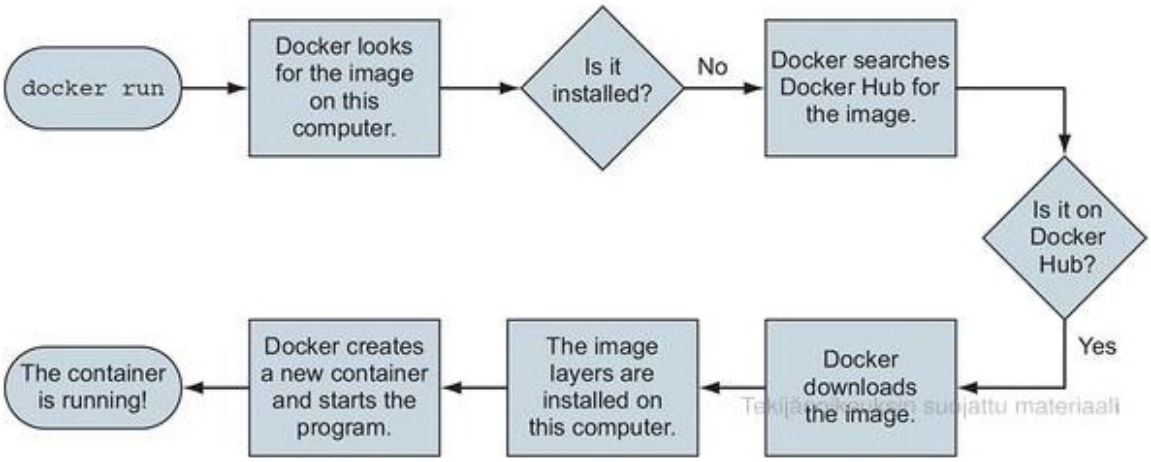
Kuvassa 6 kuvataan kaaviona Dockerin hierarkkinen malli, jossa esitellään käytettävät komennot, komponentit sekä teknologiat. Dockerissa käytettäviä komentoja ovat build, pull sekä run. Build komento suoritetaan, kun Docker-kuva luodaan paikalliseksi versioksi palvelimelta. Docker daemon työstää käyttöliittymästä tulevat komennot niin sanotuiksi kuviksi ja konteiksi, joita voidaan käyttää paikallisessa kehitysprojektissa. Dockerissa keskeisenä teknologiana ovat niin sanotut kontit, joita voidaan käyttää paikallisesti mutta jotka eivät fyysisesti sijaitse isäntäkoneella. Näihin kontteihin muodostetaan yhteys käyttöjärjestelmän portin avulla. (Waeosri 2022; Combe, Martin & Di Pietro 2016, 57).



Kuva 6. Dockerin hierarkkinen malli sekä sen käyttämät teknologiat. (Waeosri 2022).

Dockerin ohjelmisto suoritetaan käyttäjän koneella ns. daemon-tiedostona. Daemonin roolina on pakettien hallita Dockerin käyttöjärjestelmässä, jossa kontteja sekä niiden sidonnaisuuksia hallinnoidaan paikallisesti sekä palvelimella sijaitsevissa konteissa. Daemon suorittaa Dockerin ohjelmiston toiminnallisuuksia ja asennuksia komentoriviltä paikallisessa käyttöjärjestelmässä järjestelmänvalvojan käyttöoikeuksilla. (Combe ym. 2016, 57–58).

Kuvassa 7 esitellään, miten komentorivillä suoritettavassa komennossa ”Docker run” perustetaan, haetaan sisältöä tietokoneelta sekä ladataan paikalliseksi konttien sisältöä palvelimelta. Komennossa tarkastetaan samalla, mikäli vastaava Docker-kuva löytyy paikallisesti asennettuna. Palvelimelta Docker vertaa, löytyykö pyydettyä kuvaa ja lataa sen paikalliseksi, mikäli sitä ei ole vielä paikallisena. Samassa yhteydessä kuvaan liittyvät sidonnaisuudet konfiguroidaan sopimaan yhteen paikallisen projektin kanssa. (Nickoloff & Kuenzli 2019, 31–33).



Kuva 7. Docker run -komento sekä sen suorittamat toimenpiteet paikallisessa kehitysympäristössä. (Nickoloff & Kuenzli 2019, 32).

Käyttöönottoasennuksen jälkeen Dockerissa suositellaan käyttämään Dockerin omaa perehdytystä asennuksen etenemisessä, mikäli asennus tehdään käyttöjärjestelmässä ensimmäistä kertaa tai halutaan parantaa tietämystä Dockerista. Perehdytyksessä ladataan esimerkkikontti Dockerin palvelimelta paikalliseen kehitysympäristöön. Dockerin kontin asennus paikalliseksi tapahtuu käyttämällä kuvassa 8 esitettyä komentoa. (Nickoloff & Kuenzli 2019, 31–33).

```
1    sudo docker run hello-world
```

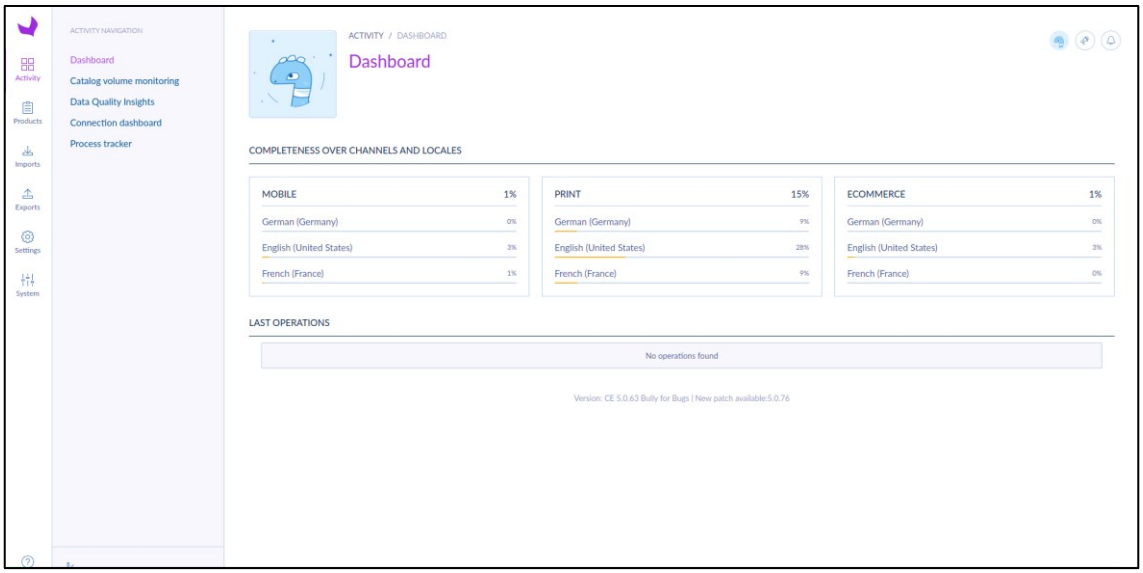
Kuva 8. Dockerin kontin käynnistämiseen liittyvä komento. (Jayanandana 2018).

Docker Hub on nimensä mukaisesti palvelimella sijaitseva Dockerin ohjelmistojen säilytyspaikka, jossa sijaitsevat esimerkiksi käyttäjien käyttämät Docker-kuvat. Käyttäjät voivat ladata Docker Hubiin omia muokkaamiaan kuvia sekä vastavuoroisesti ladata niitä paikallisesti käytettäväksi projekteissa. Lähtökohtaisesti kaikki Docker Hubissa olevat kansiot ovat julkisia. Docker Hubissa on mahdollista luoda myös omia yksityisiä kansioita, mikä on maksullinen ominaisuus. (Combe ym. 2016, 57–58).

3.4 Akeneo

Akeneossa käytetään pohjana Symfonyn paradigmaa. Tästä syystä Akeneossa ja Symfonyssa on huomattavasti samankaltaisuuksia esimerkiksi komponenteissa ja mallinnuksessa. Esimerkiksi Symfonyn komponentteja vastaavat ovat Akeneo-paketteja (bundle). Jokaisesta Akeneon projektista löytyy Symfonya mallintava kansiorakenne. Tällaisia ovat esimerkiksi controller, datagrid, dependencyinjection, doctrine jne. Symfonyn hyödyt korostuvat erityisesti edistyneemmässä ohjelmoinnissa, jossa vaaditaan nopeaa, monikanavaista ja tehokasta järjestelmää. Nämä vaatimukset ovat tavanomaisia keskisuurissa ja suurissa yrityksissä. Tämän perusteella Akeneon pohjarakenteeksi on valittu modulaarisuutta hyödyntävä Symfony. (Akeneo 2022b; Laaziri ym. 2019, 710).

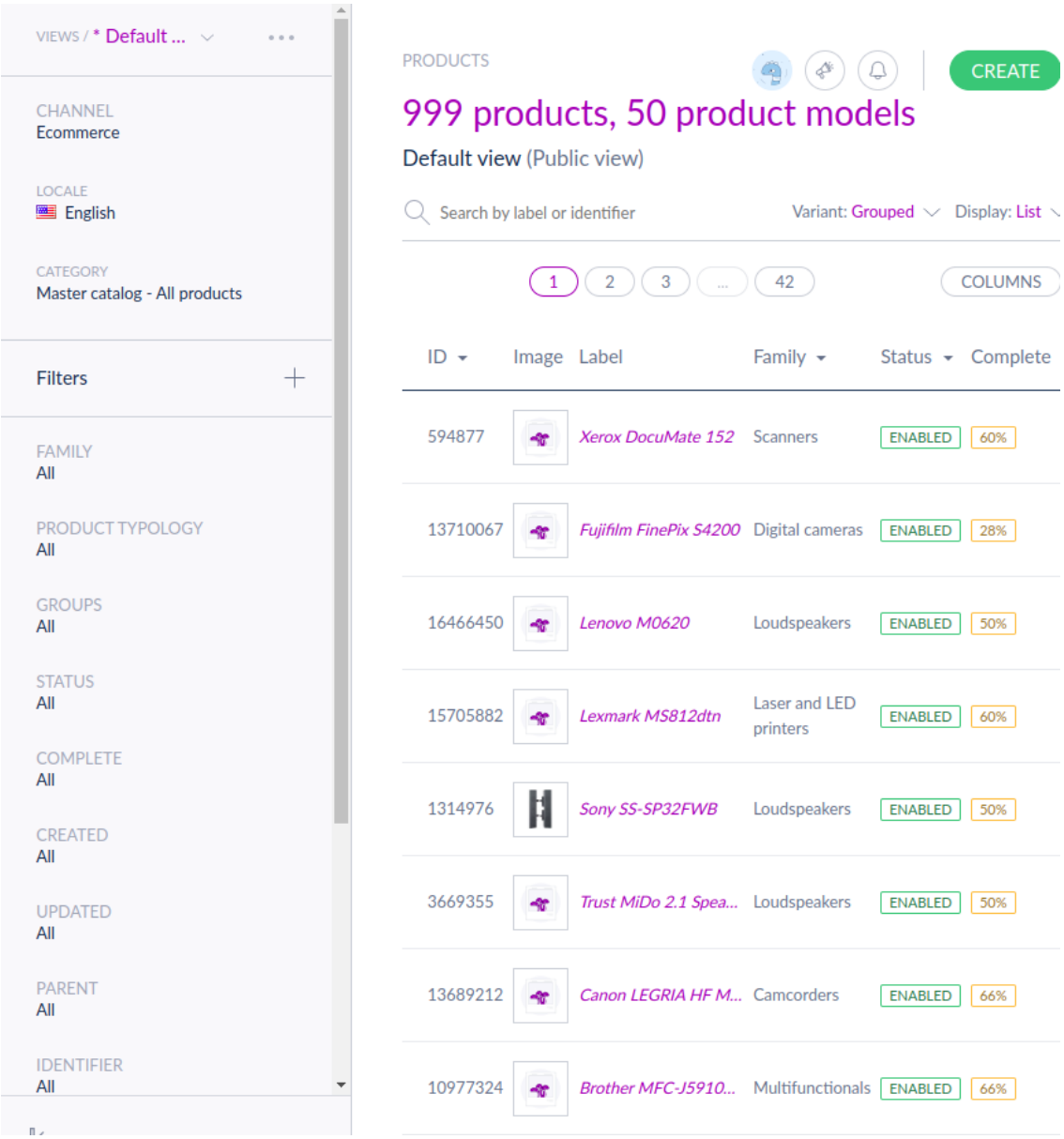
Kuvassa 9 on kuvattu Akeneon käyttöliittymän oletusnäkymä, kun sovellus käynnistyy. Yleisnäkymässä esitellään kokonaiskuvauksena Akeneossa olevat tuotteet, kategoriat, Akeneoon liitettävät kanavat sekä käynnissä olevat prosessit. Sivupalkissa olevista kuvakkeista voi siirtyä tarkastelemaan tuotteita, Akeneoon tuotavaa ja Akeneosta vietävää dataa, tuotehallinnan asetuksia sekä Akeneon järjestelmän asetuksia. Valikoissa, kuvakkeissa ja teksteissä on nähtävissä Akeneon omat tyylittelyt, joita ei ole muokattu.



Kuva 9. Akeneon oletusnäkymä sovelluksen käynnistyessä.

Kuvassa 10 on esimerkki Akeneon tuotetiedon pääkategorian yleisnäkymästä. Tuotetiedon pääkategoriassa on tuotteet, jotka on tuotu Akeneoon esimerkiksi ERP:stä. Tuotteen tietoa voidaan ri-

kastaa valitsemalla haluttu tuote kuvakkeesta tai otsikosta, jolloin aukeaa yksityiskohtainen tuotenäkymä. Tuotenäkymässä on esimerkiksi nimike, lyhyt tuotekuvaus, tuotteen sisältämät kuvat, tuotteen tyyppi, paino, varastosaatavuus jne. Kategoriassa näytettäviä tuotteita voi rajata sivupalkissa näkyvillä filttereillä, eli attribuuteilla.



Kuva 10. Akeneon tuotetiedon rikastamisen kategorianäkymä.

3.5 Muut paikallisen kehittämisen työkalut

PHPstorm on tämän opinnäytetyön toteutuksessa käytetty ohjelmointieditori. PHPStorm sisältää erityisesti PHP:lla ohjelmoitaessa helpottavaa muokattavuutta, kuten koodin formatointia ja korostamista, eri ohjelmointikielien tuen sisällyttämistä oletusarvoisena, virheiden lukutaitoa PHP:n syntaksointiin verraten, kirjanmerkkien luomisen halutuille koodin osille sekä monipuolista koodin muokattavuutta kehittäjän omien mieltymysten mukaisesti. PHPStorm on JetBrains:n kehittämä ja ylläpitämä maksullinen ohjelmointieditori. (Chaudhary & Kumar 2014, 12).

Symfonylle on lisäksi tehty PHPStormiin oma tuki, jolla kehitysympäristön yhteyksien ajantasaisuutta voi helposti seurata sekä päivittää tarvittaessa. Lisäosa tunnistaa Symfonyssa käytettävän logiikan ja syntaksoinnin, jonka pohjalta se tarvittaessa ehdottaa automaattista koodin viimeistelyä tai kytkemistä yleisesti käytettäviin Symfonyn ratkaisuihin. (Chaudhary & Kumar 2014, 112–114).

Versionhallinnan käyttäminen ohjelmoinnissa on keskeisessä asemassa, sillä suurin osa tehdystä kehittämistyöstä globaalisti tapahtuu suurelta osin edelleen paikallisesti kehittäjän kehitysympäristössä. Kehittämistyön tulee olla saatavilla samanaikaisesti muille projektiin osallistuville, jotta tehtävät muutokset ovat linjassa kokonaisuuteen nähden sekä asetettujen tavoitteiden täyttämiseksi. Git on paikallisessa kehittämisessä käytettävä versionhallinnan työkalu, joka tarjoaa kilpailijoihinsa nähden nopeaa tiedostojen siirtoa, riippumattomuutta jatkuvasta verkkoyhteydestä sekä tilan minimaalista käyttöä. (Chacon & Straub 2014, 10–11).

Git:in käyttöön liittyvät hyödyt korostuvat erityisesti ryhmätöissä. Git vertaa kehitystyön eri versioiden muutoksien mahdolliset konfliktit ja päällekkäisyydet koodissa etukäteen ennen kuin versioita ollaan yhdistämässä samaan jaettuun työympäristöön. Tällä tavalla voidaan välttää ajallista menetystä sekä ylimääräistä resurssien hukkaamista inhimillisten virheiden vuoksi. (Chacon & Straub 2014, 11).

Tässä opinnäytetyössä tehdyt tyylitiedostojen muokkaukset tulevat muidenkin kehittäjien hyödynnettäviksi ja jatkokehitettäväksi, ja niitä voidaan lisätä tulevaisuudessa muihinkin projekteihin. Tässä opinnäytetyössä tehty kehitystyön tulos ladattiin Git:ä käyttämällä Pinjan ylläpitämälle palvelimelle.

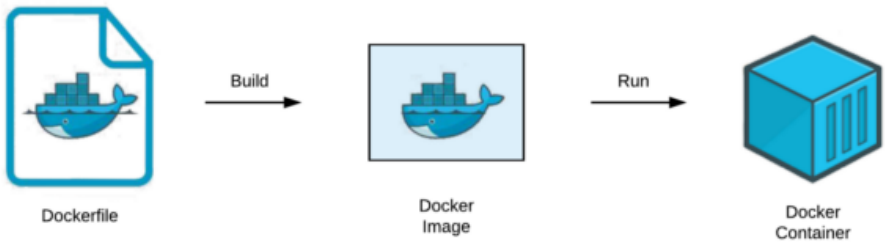
4 PAIKALLISEN KEHITTÄMISTYÖN TYÖKALUT

4.1 Dockerin asennus ja käyttöönotto

Asennuksessa ensimmäisenä Docker ladataan sen omalta verkkosivulta sekä suoritetaan käyttöjärjestelmässä käyttöönoton asennus. Dockerin asennuksessa valitaan sopiva versio käytössä olevan käyttöjärjestelmän perusteella. Windowsille, Mac:lle sekä Linuxille on olemassa omat asennusohjelmansa. Muille käyttöjärjestelmille löytyy Dockerin sivuilta vastaavasti ohjeet yksityiskohtaiseen asentamiseen ja toiminnan testaamiseen. (Explore Docker's Container Image Repository | Docker Hub 2022).

Dockerin asennuksessa on suositeltavaa käyttää viimeisintä julkaistua vakaata versiota. Dockerin versioita ja muita tiettyjä virallisia julkaisuja on mahdollista myös asentaa tilannekohtaisesti komentoriviltä. Tilanne voi olla esimerkiksi, että projekti on luotu aiemmin tietylle versiolle, jolloin yhteensopivuus on otettava huomioon. Asennusohjeet löytyvät Dockerin omilta verkkosivuilta asennuksen aloituksen yhteydessä. Dockerin asentamisessa tärkein huomio on asennuksen jälkeinen tarkastus, että Docker käynnistyy ja tiedonsiirrossa ei esiinny virheitä.

Docker käyttää kuvien asennukseen ja määrittämiseen niin sanottuja Dockerfileja. Dockerfile on yksinkertainen tekstitiedosto, missä määritetään, miten Dockerin luomassa kuvassa tarvittavia resursseja hyödynnetään projektissa. Dockerfilessä määritellään esimerkiksi projektin ympäristön käyttämät kansiot sekä tarvittavat liitännäiset ohjelmat. Kuvassa 11 Dockerfile luodaan, joka toimii kuvien käsittelemisen sekä konttien hyödyntämisessä paikallisen kehittämisen perustana. (Jayanandana 2018).



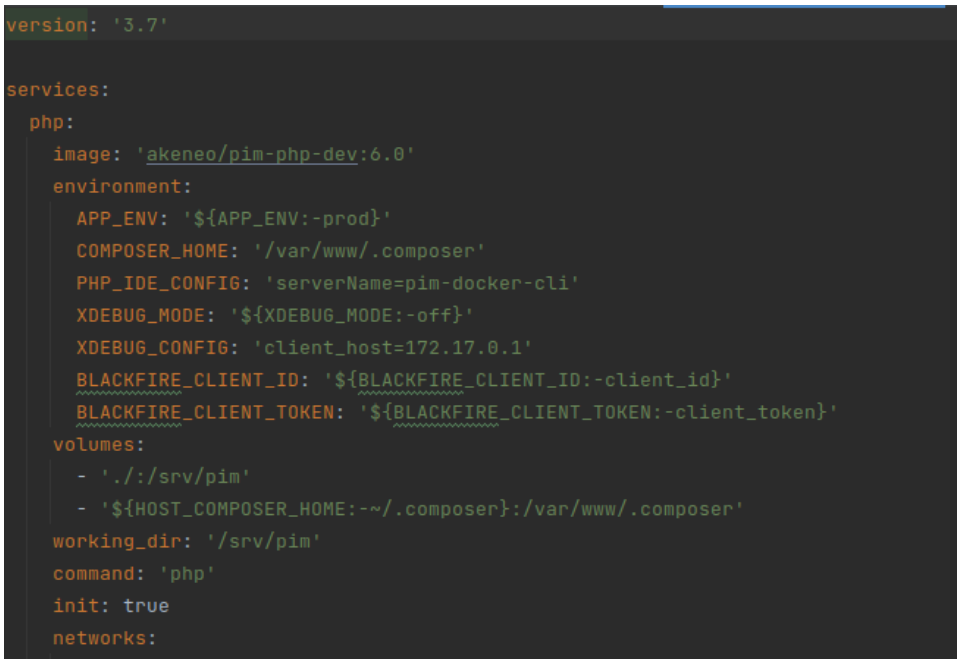
Kuva 11. Dockerfilen käyttö kuvien lataamisessa sekä konttien hyödyntämisestä paikallisessa projektissa. (Jayanandana 2018).

Kuvassa 12 on esimerkki yksinkertaisesta Dockerfilestä, joka suoritetaan käyttämällä komentoa docker run. Dockerfile voi esimerkiksi sisältää tietoa tiedoston luojasta, Dockerin käyttämästä versiosta sekä Dockerin varaamasta paikallisen koneen käyttämästä portista. Onnistuneessa asennuksessa, oheinen komento lataa paikalliseksi ”hello-world” testikontin, joka käynnistää, tulostaa tekstin sekä viimeisenä sulkee kontin.

```
# Version: 0.0.1
FROM ubuntu:14.04
MAINTAINER James Turnbull "james@example.com"
RUN apt-get update
RUN apt-get install -y nginx
RUN echo 'Hi, I am in your container' \
    >/usr/share/nginx/html/index.html
EXPOSE 80
```

Kuva 12. Yksinkertainen Dockerfile jossa määritetään luotavan Docker-kuvan sisältö paikallisessa projektissa. (Turnbull 2014, 102).

Kuvassa 13 esitellään Dockerfileen pohjautuvaa docker-compose tiedostoa esimerkin mukaisesti. Docker-compose tiedostossa määritellään, miten Akeneo projektissa Dockerin resursseja paikallisessa kehittämisessä hyödynnetään. Docker-compose muun muassa hallinnoi tarvittavaa kansioiden käyttöoikeustasoa sekä ylläpitää projektiin sidonnaisia ympäristöjä ajan tasalla.



```yaml
version: '3.7'

services:
  php:
    image: 'akeneo/pim-php-dev:6.0'
    environment:
      APP_ENV: '${APP_ENV:-prod}'
      COMPOSER_HOME: '/var/www/.composer'
      PHP_IDE_CONFIG: 'serverName=pim-docker-cli'
      XDEBUG_MODE: '${XDEBUG_MODE:-off}'
      XDEBUG_CONFIG: 'client_host=172.17.0.1'
      BLACKFIRE_CLIENT_ID: '${BLACKFIRE_CLIENT_ID:-client_id}'
      BLACKFIRE_CLIENT_TOKEN: '${BLACKFIRE_CLIENT_TOKEN:-client_token}'
    volumes:
      - './:/srv/pim'
      - '${HOST_COMPOSER_HOME:-~/.composer}:/var/www/.composer'
    working_dir: '/srv/pim'
    command: 'php'
    init: true
    networks:
```

Kuva 13. Dockerin käyttämä docker-compose tiedosto sekä esimerkki Akeneo-projektin ympäristön määrityksistä.

4.2 Akeneon asennus ja käyttöönotto

Kehitystyö aloitetaan asentamalla projektiin soveltuva Akeneo-versio paikallisesti omalle koneelle. Akeneon asentamisessa paikalliseksi projektiksi on suositeltavaa käyttää esimerkiksi Akeneon omilta sivuilta löytyvää ohjetta, jotta asentaminen onnistuisi mahdollisimman vaivattomasti. Akeneon Community Edition on saatavilla ilmaiseksi GitHubissa. Version valitsemisessa on hyvä huomioida, että uudemmat versiot eivät ole suoraviivaisesti taaksepäin yhteensopivia. Tämän osalta olisi tärkeää tietää, mikäli tarkoituksena on toteuttaa olemassa olevaan projektiin uusia ominaisuuksia, mitä versiota tullaan käyttämään projektissa. (Akeneo 2022c).

Community Edition on ilmainen versio, jota käytetään yleisesti kehitysympäristönä kehittäjien keskuudessa eri toiminnallisuuksien testaamista ja kehittämistä varten. Tässä opinnäytetyössä käytettävä versio on Community Edition. Vaihtoehtoisesti Akeneosta on olemassa myös yritysversio eli Enterprise Edition. (Akeneo 2022c).

Linuxille paikallista projektia asennettaessa käytetään Unix-pohjaisia komentoja. Kuvan 14 mukaisesti Akeneon asentamisen ensimmäisessä vaiheessa luodaan ensin uusi, tyhjä kansio valittuun hakemistoon sekä siirrytään siihen. Kansion nimellä ei ole vaikutusta projektin käytettävyyteen. On

kuitenkin huomioitava, että asennusta jatketaan siitä hetkestä lähtien valittua nimeä käyttäen. Esimerkin vuoksi käytämme tässä yhteydessä ohjeessa mainittua kansion nimeä, joka on nimeltään pim. (Akeneo 2022c).

```
mkdir pim
cd pim
```

Kuva 14. Tyhjän kansion luominen tulevaa Akeneo-projektia varten.

Akeneon asentaminen paikalliseksi voidaan toteuttaa kahdella eri tavalla. Asennuksessa voidaan käyttää Dockeria tai asentamisen voi toteuttaa myös ilman Dockeria. Asennustavassa, jossa ei käytetä Dockeria, kehittäjän tulee itse asettaa oman käyttöjärjestelmänsä asetukset manuaalisesti Akeneon kanssa yhteensopiviksi. Tässä opinnäytetyössä seurataan Dockerin käyttämiseen perustuvaa ohjetta. Akeneon asentamisessa paikalliseksi projektiksi käyttäen Dockeria komentoriviltä (Linuxissa vastaavaa kutsutaan terminaaliksi) suoritetaan oheisen kuvan 15 mukainen komento.

```
$ docker run -ti -u www-data --rm \
    -e COMPOSER_MEMORY_LIMIT=4G \
    -v $(pwd):/srv/pim -v ~/.composer:/var/www/.composer -w /srv/pim \
    akeneo/pim-php-dev:5.0 php /usr/local/bin/composer create-project \
    akeneo/pim-community-standard /srv/pim "5.0.*@stable"
```

Kuva 15. Akeneon asentaminen paikalliseksi kehitysympäristöön käyttämällä Dockerin konttiteknologiaa.

Erityistä huomiota on kiinnitettävä ennen komennon suorittamista, että käyttöoikeudet asetetaan oikein uutta Akeneo-käyttäjää varten Linuxissa. Akeneo käyttää Linuxissa Dockeria asentamisessa niin kutsuttua www-data-käyttäjää, jolloin sillä on oltava kansiossa riittävän laajat käyttöoikeudet toimiakseen. Mikäli kansion käyttöoikeus ei ole riittävän laaja www-data-käyttäjää varten, projektista tulee todennäköisesti käyttökelvoton. Yksi vaihtoehto käyttöoikeuksien muokkaamiseen on määrittää Dockerfilessa Dockerille niin sanotut laajat UID ja GID-oikeudet. Muussa tapauksessa Docker ei pysty käyttämään paikallisesti palvelimelta ladattuja kontteja projektissa. Komennossa

ladataan Akeneosta 5.0 versio, joka oli opinnäytetyön teon hetkellä vakaa julkaisu versio. (Akeneo 2022c).

Komennossa Dockerin avulla varataan aiemmin luotu tyhjä kansio Akeneon käyttöön eli niin sanotusti mountataan. Docker lataa Akeneon palvelimelta konttien sisällön www-data-käyttäjänä sekä perustaa yhteisesti projektin selaimelle jaetun kansion /srv/pim. Paikallisessa ohjelmoinnissa kehittäjän laitteella on oltava myös version- ja -pakettienhallinnan työkalut asennettuna, kuten Composer pakettienhallintaan sekä Git, GitHub tai vastaava versioiden hallinnan työkalu. Tietokoneen tehokkuuden mukaan asentamisessa paikalliseksi kuluu komennon suorittamisesta noin 5–15 minuuttia onnistuneessa asennuksessa. (Akeneo 2022c).

Dockerin perustettua projekti onnistuneesti koneelle, projekti käynnistetään ensimmäisen kerran Unix-laitteilla käyttämällä komentorivillä komentoa make. Komento make viittaa tässä yhteydessä tiedostoon nimeltään Makefile. Makefile käynnistää useita eri Akeneo projektissa tarvittavia palveluita, joita ovat esimerkiksi PHP, Mysql, Node jne. Kuvassa 16 Akeneon paikallisen projektin sammuttamisen ja konttien poistamisen suorittavassa komennossa listautuu muun muassa ne palvelut, joita kehitystyössä hyödynnetään.

.

```
leinmik@pii-leinmik:~/Public/git/pim/pim-community-standard$ sudo make down
docker-compose down -v
Stopping pim-community-standard_httpd_1           ... done
Stopping pim-community-standard_selenium_1        ... done
Stopping pim-community-standard_pubsub-emulator_1 ... done
Stopping pim-community-standard_fpm_1             ... done
Stopping pim-community-standard_object-storage_1  ... done
Stopping pim-community-standard_mysql_1           ... done
Stopping pim-community-standard_elasticsearch_1   ... done
Removing pim-community-standard_httpd_1           ... done
Removing pim-community-standard_selenium_1        ... done
Removing pim-community-standard_node_1            ... done
Removing pim-community-standard_pubsub-emulator_1 ... done
Removing pim-community-standard_fpm_1             ... done
Removing pim-community-standard_object-storage_1  ... done
Removing pim-community-standard_php_1             ... done
Removing pim-community-standard_mysql_1           ... done
Removing pim-community-standard_blackfire_1       ... done
Removing pim-community-standard_elasticsearch_1   ... done
Removing network pim-community-standard_pim
```

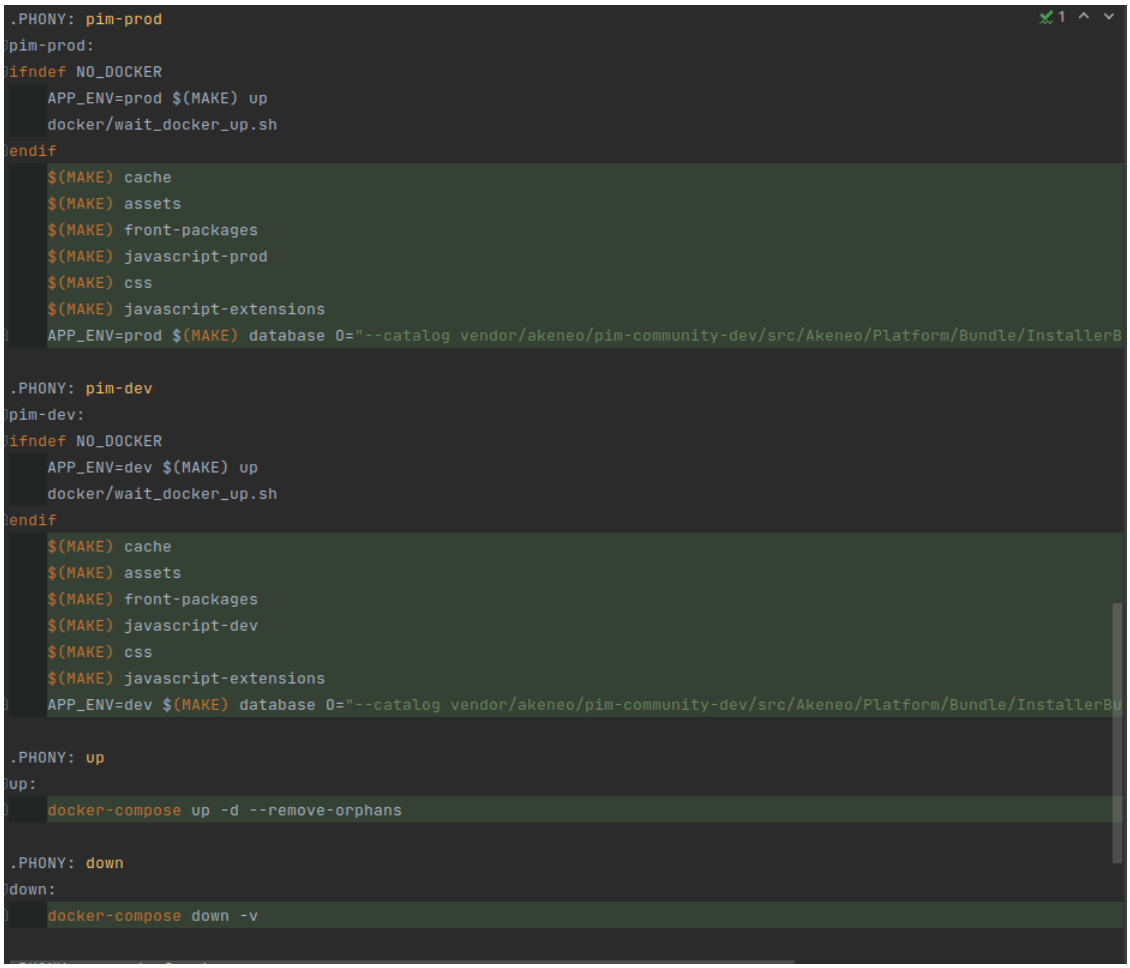
Kuva 16. Dockerin kontit, joita Akeneo käyttää sovelluksen suorittamisessa.

Akeneon kehitysympäristön perustamiseen Linuxilla tarvitaan Makefileä, jossa käytetään esimerkiksi Dockeria projektin suorittamiseen. Makefilessa määritellään, mitä resursseja projektissa suoritetaan. Kuvassa 17 on esitetelty paikallisen projektin Makefilen sisältöä. Projektin makefilessä esimerkiksi käynnistetään Docker, PHP:n liitännäisyydet sekä Yarn. Yarn on ohjelmistopakettien hallinnan työkalu.

```
#
# This file is a template Makefile. Some targets are presented here as examples.
# Feel free to customize it to your needs!
#
CMD_ON_PROJECT = docker-compose run -u www-data --rm php
PHP_RUN = $(CMD_ON_PROJECT) php
YARN_RUN = docker-compose run -u node --rm -e YARN_REGISTRY -e PUPPETEER_SKIP_CHROMIUM_DOWNLOAD node yarn

ifdef NO_DOCKER
  CMD_ON_PROJECT =
  YARN_RUN = yarnpkg
  PHP_RUN = php
endif

.DEFAULT_GOAL := dev

yarn.lock: package.json
    PUPPETEER_SKIP_CHROMIUM_DOWNLOAD=1 $(YARN_RUN) install

node_modules: yarn.lock
    PUPPETEER_SKIP_CHROMIUM_DOWNLOAD=1 $(YARN_RUN) install

.PHONY: javascript-extensions
javascript-extensions:
    $(YARN_RUN) run update-extensions

.PHONY: front-packages
front-packages:
    PUPPETEER_SKIP_CHROMIUM_DOWNLOAD=1 $(YARN_RUN) packages:build
```

Kuva 17. Makefilessä määritellyt paikallisen kehitysympäristön asetukset.

Makefilestä voidaan tarvittaessa suorittaa vain ne palvelut, joita tarvitaan käynnistämättä kaikkia palveluita. Tällainen tilanne voi esimerkiksi olla, jos halutaan testata Akeneon käyttäytymistä kehitysympäristössä tai vastaavasti tuotannossa. Akeneon käynnistäminen kehitysympäristössä tapahtuu käyttämällä komentoa make pim-dev tai pim-prod. Näiden kahden palvelun eroja on esitelty kuvassa 18. Merkittävimmät erot palveluiden välillä liittyvät siihen, miten tietokanta sekä selaimen käyttäytämät prosessit käyttäytyvät suorittamisen aikana.



```
.PHONY: pim-prod
pim-prod:
ifndef NO_DOCKER
    APP_ENV=prod $(MAKE) up
    docker/wait_docker_up.sh
endif
    $(MAKE) cache
    $(MAKE) assets
    $(MAKE) front-packages
    $(MAKE) javascript-prod
    $(MAKE) css
    $(MAKE) javascript-extensions
    APP_ENV=prod $(MAKE) database O="--catalog vendor/akeneo/pim-community-dev/src/Akeneo/Platform/Bundle/InstallerB

.PHONY: pim-dev
pim-dev:
ifndef NO_DOCKER
    APP_ENV=dev $(MAKE) up
    docker/wait_docker_up.sh
endif
    $(MAKE) cache
    $(MAKE) assets
    $(MAKE) front-packages
    $(MAKE) javascript-dev
    $(MAKE) css
    $(MAKE) javascript-extensions
    APP_ENV=dev $(MAKE) database O="--catalog vendor/akeneo/pim-community-dev/src/Akeneo/Platform/Bundle/InstallerBu

.PHONY: up
up:
    docker-compose up -d --remove-orphans

.PHONY: down
down:
    docker-compose down -v
```

Kuva 18. Makefilessä määritellyt eri toiminnallisuuksien käynnistämiseen tarkoitetut komennot, jotka voidaan suorittaa komentoriviltä.

5 KEHITTÄMISTYÖN TOTEUTUS

5.1 Käyttöliittymän ulkoasun yksilöimisen suunnittelu

Opinnäytetyön aihe on lähtöisin tarpeesta, jossa yhteistyöyritykset voisivat käyttää Akeneon tarjoamia palveluita entistä paremmin yrityksen tunnistettavuuden lisäämiseksi. Akeneon käyttöliittymän yksilöimisen tarkoituksena on osoittaa käyttäjälleen, että yritys hyödyntää omassa toiminnassaan tuotetiedonhallinnan järjestelmää. Käyttöliittymän ulkoasu vaikuttaa käyttäjän mielikuvaan sovelluksesta, mikä vaikuttaa vuorostaan siihen, miten käyttäjä sitoutuu hyödyntämään yrityksen palveluita (Sparks, Perkins & Buckley 2013, 8–9).

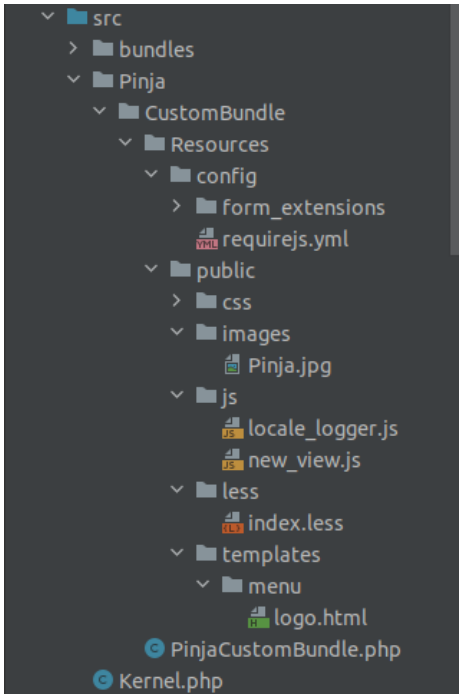
Pinjalla käyttöliittymän kehittämiseksi tarkoitetut materiaalit toimittivat käytettäväkseni yksikössä työskentelevä graafinen suunnittelija. Materiaaleihin sisältyi muun muassa logoja, kuvioita (painikkeet) sekä ennalta valitut RGB-värisävyt. Graafisen ulkoasun suunnittelemisessa Pinjan edellytyksenä oli, että käyttöliittymän tulisi olla yhdistettävissä Pinjan yrityksen mukaiseen ulkoasuun. Ulkoasun tulisi olla lisäksi tulevia asiakasprojekteja ajatellen helposti muokattavissa. Tulevaisuudessa käyttöliittymän yksilöiminen asiakkaiden tarpeisiin tulisi tapahtua mahdollisimman vaivattomasti. Suunnittelun aikana konsultoitiin Pinjan Akeneo-projektien integraatiospesialistia, jonka näkemykset ja kokemus Akeneon käytettävyydestä olivat arvokkaita suunnannäyttäjiä kokonaisuuden suunnittelussa. Pinjan graafisena suunnittelijana toimineelta henkilöltä saatiin kuvankäsittelyn ohjelman avulla teemaan sovitettuja kuvakkeita, painikkeita, värikartta sekä ohjeita valmiiden ratkaisujen hyödyntämiseen.

Akeneon käyttöliittymän yksilöimisen suunnittelussa hyödynnettiin myös Pinjassa aiemmin käytössä olleita moduuleita. Pinjan käytössä olevien moduulien yhdistämisellä saatiin luotua lopullinen käyttöliittymän ulkoasu, joka on yhtenäinen muiden Pinjan projektien kanssa. Sovelluksen ulkoasun toteutus on sekä Akeneon että Pinjan moduulien yhdistämisen lopputulos.

5.2 Käyttöliittymän yksilöimisessä tehdyt toteutukset

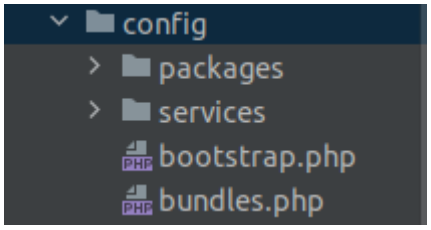
Tässä opinnäytetyössä Akeneon käyttöliittymän yksilöinnissä on seurattu Akeneon omaa ohjeistusta. Akeneon sivustolla on yleiset ohjeet käyttöliittymän muokkaamiseen, joiden avulla luotiin käytännön toteutus. Akeneon ohjeita seuraamalla sovelluksen yksilöity käyttöliittymä saatiin päivittymään selaimessa. Opinnäytetyössä seurattiin Akeneon version 5.0 pohjalta laadittua asennusohjetta. (Akeneo 2022d). Akeneon ohjeistuksen lisäksi konsultoitiin myös kehittäjää, joka käyttää Akeneota useissa kehittämistyön projekteissa.

Kansiorakenteessa noudatettiin Pinjan käytännön ohjeistusta, jossa tavoiteltiin selkeyttä ja johdonmukaisuutta. Kansioiden ja luokkien nimeämisessä tärkeää on se, että ne kuvaavat käyttötarkoitustaan. Linjauksen mukaan kansainvälisessä työympäristössä kaikkien tulisi saada nopeasti käsitys luokkien sijainnin ja nimeämisen perusteella yleiskäsitys moduulin sisällöstä ja käyttötarkoituksesta. Linjauksella nopeutetaan aiemmin projektissa työskentelemättömien työntekijöiden perehtymistä projektin keskeiseen sisältöön. Uusi moduuli on toteutettu src-kansion alle, joka sijaitsee projektin juurikansiossa. Kuvassa 19 on esiteltynä käyttöliittymän muokkaamiseen liittyvän moduulin kansiorakenne.



Kuva 19. Akeneon käyttöliittymän ulkoasuun vaikuttavan moduulin kansiorakenne projektin juuressa.

Käyttöliittymän muokkaamiseksi tulee Akeneossa perustaa kuvan 20 mukainen yleinen bundle-luokka, joka ei sijaitse projektissa käytettävässä Pinjan moduulin kansiossa. Kansio sijaitsee projektin juuressa config-kansiossa. Perustettu luokka otetaan käyttöön Akeneo-järjestelmässä määrittelemällä se bundles.php-tiedostossa kuvan 21 kuvaamalla tavalla.



Kuva 20. Akeneon bundle-luokka sijaitsee projektin juuressa.

```php
<?php

return [
    // Add your bundles here with the associated env.
    // Ex:
    // Acme\Bundle\AppBundle\AcmeAppBundle::class => ['dev' => true, 'test' => true, 'prod' => true]
    Pinja\CustomBundle\PinjaCustomBundle::class => ['all' => true]
];
```

Kuva 21. Tarvittavat määritykset bundle-luokan käyttöönottamiseksi Akeneossa.

Akeneon käyttöliittymän muokkaamisessa tarvitaan funktiota, joka hyödyntää UserContext-moduulin toiminnallisuuksia. UserContext hyödyntää katalogin sisältöä käyttämällä luotua objektia. Kyseisen toiminnallisuus hyödyntää Akeneon käyttöliittymää ohjaavaa moduulia, jolle kerrotaan, mitä uusia toiminnallisuuksia Pinjan moduulista lisätään siihen. Kuvassa 22 kuvataan tätä toiminnallisuuden määrittelemistä koodissa.

```
1   define(
2       ['pim/user-context'],
3       function (UserContext) {
4           return {
5               logLocale: function () {
6                   console.log(UserContext.get('uiLocale'));
7               }
8           };
9       }
10  );
```

Kuva 22. Akeneon käyttöliittymän moduuliin lisätään hyödyntämään Pinjan moduulin sisältöä.

Moduulin nimeämisessä on valittu käytettäväksi nimeä PinjaCustomBundle, joka kuvastaa nimellään käyttöliittymän muokkausta. Moduulin perustamisessa ja käyttöönottamisessa luodaan kyseisestä moduulista bundle, jotta Akeneo käyttää sitä muokatun käyttöliittymän kokoelmana. PinjaCustomBundle perii ominaisuutensa Akeneon superluokka-bundlelta, jonka tietoa käsitellään kokoelmana erilaisia käsittelijöitä ja templaatteja (Salehi 2016, 16–17). Kuvassa 23 kuvataan bundlen määrittelemistä projektissa.

```php
<?php

namespace Pinja\CustomBundle;

use Symfony\Component\HttpKernel\Bundle\Bundle;

/**
 * Class PiimegaSimpleBundleProductBundle
 * @package PiimegaSimpleBundleProductBundle
 */
class PinjaCustomBundle extends Bundle
{

}
```

Kuva 23. Projektissa käytettävä käyttöliittymän moduuli hyödyntää Symfonyssa määritettyä superluokka-bundlea.

Pinjan moduuli on rekisteröitävä RequireJS:ää hyödyntävässä konfiguraatiotiedostossa. Konfiguraatio on nimeltään my_locale_logger, joka rekisteröidään järjestelmän käyttöön. Tiedoston lisäosan julistamista ei tarvitse tehdä erikseen projektissa, sillä se kuuluu osaksi RequireJS:n toiminnallisuuksia. Tiedosto on siten järjestelmän löydettävissä public web-kansiossa, jossa sijaitsee muun muassa sovelluksen selaimeen päivitettävät tiedot. Tiedon siirto public-kansioon tapahtuu samassa yhteydessä, kun Akeneon assetit asennetaan komentoa käyttämällä. Konfguraation voi nimeämisellä tulevaisuudessa käytettävissä projekteissa ei ole vaikutusta. Tiedostossa määritetään samalla järjestelmälle, missä käyttöliittymän yksilöity logo sijaitsee. Kuvassa 24 kuvataan konfiguraatiotiedosto määrityksiä.

```
config:
  paths:
    my_locale_logger: pinjacustom/js/locale_logger
    pim/template/menu/logo:     pinjacustom/templates/menu/logo.html
```

Kuva 24. Pinjan moduulin rekisteröiminen sekä logon asettaminen käyttämällä RequireJS konfiguraatiota.

Uuden moduulin käyttöönotossa suoritetaan sarja komentoja komentoriviltä, joilla vaikutetaan Akeneon eri palveluiden rekisteröimiseen ja päivittämiseen. Näihin toimintoihin kuuluvat esimerkiksi välimuistin tyhjentäminen, assettien rekisteröiminen web-kansioon, css-tiedostojen koonti käyttämällä yarnia (package manager) sekä lopuksi bundle.js tiedoston rakentamista edellä mainituista muutoksista. Viimeisimmässä komennossa kuvan 25 mukaisesti koostetaan ja minimoidaan kaikki pim-tiedostot web-tiedostoiksi sekä viedään public-kansioon. Tämän viimeisen komennon jälkeen tehdyt muutokset moduuliin ja sovelluksen käyttöliittymään on päivitetty ajan tasalle (Akeneo 2022d).

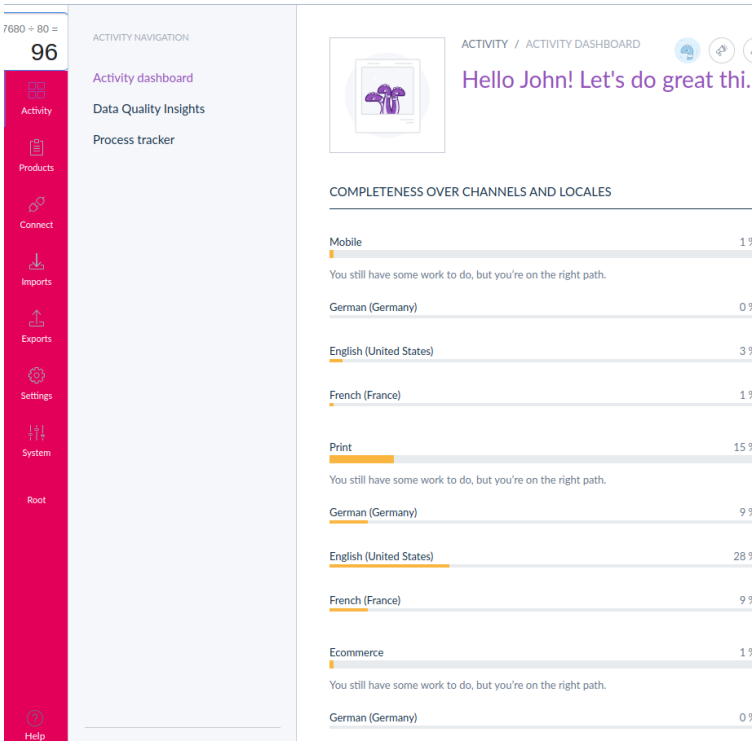
```
yarn run webpack
```

```
rm -rf ./var/cache/*
```

```
bin/console assets:install web
```

Kuva 25. Paikallisessa kehitysympäristössä suoritettavat komennot, joilla moduulissa tehtävät muutokset saadaan päivitettyä sovelluksesta selaimeen.

Kuvassa 26 havainnollistetaan, miten aiempi moduulin rekisteröiminen, RequireJS tiedoston määritykset sekä kehitysympäristössä suoritettavat komennot saavat moduuliin näyttämään selaimessa. Käyttöliittymän muotoilussa tunnistettavin vaihe on, miten Akeneo käsittelee projektissa olevia uusia moduuleja. Uuden luodun moduulin sisällön tulee yli kirjoittaa alkuperäisesti Akeneo-projektissa olevien moduulien sisältö, jotta tehdyt muutokset päivittyvät selaimessa.



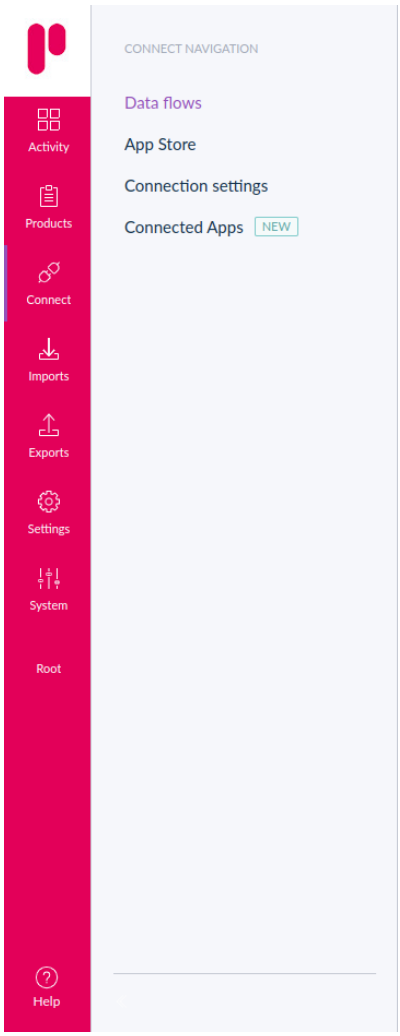
Kuva 26. Käyttöliittymän kuvakkeiden ja värityksien testaamista selaimessa muutoksien päivittämisen jälkeen.

Värien muokkaamiset ja logon muut määritykset toteutetaan less-tyylitiedostoilla (Learner Style Sheets). Less-tiedostot ovat pohjimmiltaan kuten CSS-tyylitiedostot mutta sen lisäksi se myös sisältää integroitua JavaScript kieltä. Less erottuu CSS:stä selvimmin sillä, että siinä käytetään pistettä elementin kohdistamiseen. Mikäli kohdistus tarvitsee tehdä elementin lapsielementteihin, silloin käytetään aritmeettista osoitinta suurempi kuin kohdentamaan haluttuun elementtiin. Akeneon käyttöliittymän näkyvimmät muutokset on toteutettu index.less-tiedostolla, jolla voidaan muun muassa vaikuttaa logoon ja tekstien väritykseen. (Getting started | Less.js 2022) .

Kuvassa 27 kuvataan less-tiedostoon tehtyjä määrityksiä, joilla käyttöliittymän värien muotoilu on toteutettu moduulissa. Kuvassa 28 havainnollistetaan lopputulosta kaikista opinnäytetyössä tehdyistä toteutuksista, joilla kehitystehtävä on toteutettu.

```
.AknHeader > nav > div:first-child {
  background: #e30059;
}

.AknHeader > nav > div > div:nth-child(2) > a {
  color: white;
}

.AknHeader > nav > div > div:nth-child(3) a {
  color: white;
}

.AknHeader > nav > div svg {
  color: white;
}

.AknHeader > nav > div .AknHeader-menuItemImage--iconHelp {
  filter: brightness(1000%);
}
```

Kuva 27. Less-tiedostoon tehdyt määritykset, joilla värit saadaan asetettua käyttöliittymän sivuvalikkoon.

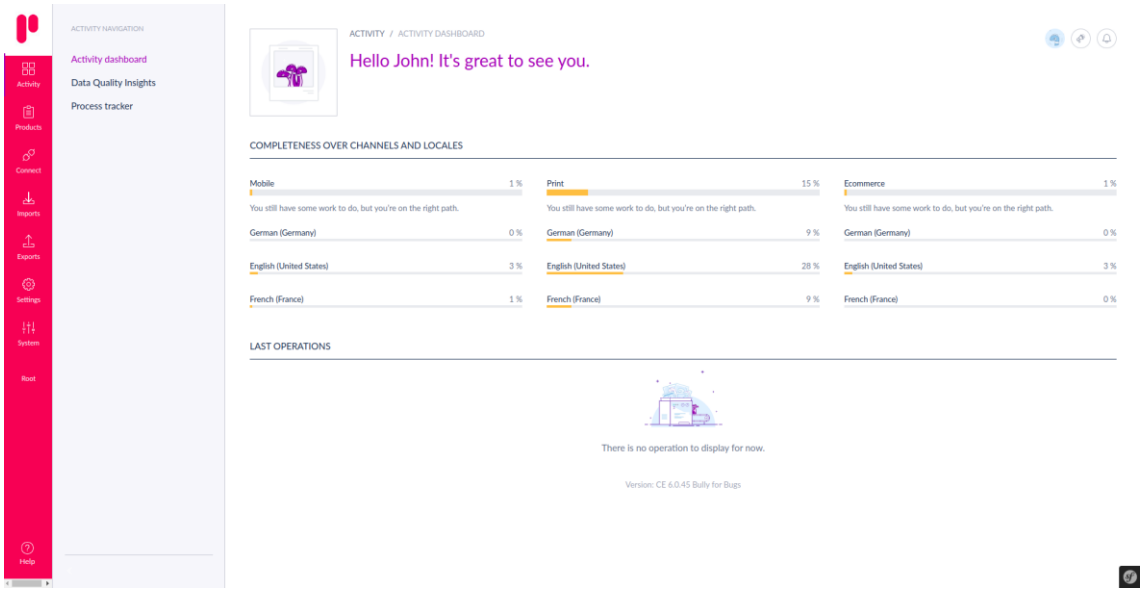


Kuva 28. Kehitystyön lopputulos selaimessa viimeisten muutosten päivittämisen jälkeen.

5.3 Paikallisen kehittämistyön työn tulokset

Yritykselle kuluttajamarkkinoilla kilpailuvaltteina toimivat sovellusten erottuvuus ja helppokäyttöisyys, joiden perusteella käyttäjät muodostavat mielikuvan. Käyttäjän mielikuva sovelluksesta on yhteydessä pidemmällä aikavälillä myös yrityksen imagoon yleisemmin. (Sparks ym. 2013, 6–7). Visuaalinen ilme auttaa jäsentämään ja hahmottamaan ohjelmiston käytettävyyttä, kun painikkeet ovat selkeästi erotettavissa taustasta (Sparks ym. 2013, 21–22).

Käyttöliittymän tyylittelyllä tuodaan näkyväksi Pinjan ominaisvärit sivupalkissa. Pinjan yrityksen logon lisääminen sekä värityksen näkyvyys sivupalkissa auttavat käyttäjää yhdistämään tarjotut tuotehallinnan palvelut helpommin, mikä vahvistaa pidemmällä aikavälillä Pinjan mieleenpainuvuutta sekä tuottoisuutta. Kuvan 29 mukainen käyttöliittymä yhdistää yrityksen ja Akeneon yhteistyön mielikuvan markkinoinnin työkaluksi.



Kuva 29. Akeneon kehittämistehtävän tuloksena valmistunut käyttöliittymä Pinjan käyttöön.

5.4 Jatkokehittämiskohteet

Tässä opinnäytetyössä käyttöliittymä yksilöitiin käyttämällä Pinjan omaa suunnitelmaa ulkoasua konseptina. Kehittämistyönä toteutettiin Akeneon aloitusnäkymässä päävalikon logon muokkaaminen sekä valikoiden pohjavärin sekä tekstien muutos. Kehittämistä valmistuneen opinnäytetyön myötä on mahdollista jatkaa Akeneossa useammalla tavalla. Mahdollisuuksia toteutetun kehitystyön pohjalta olisi esimerkiksi jatkaa tyylittelyihin syventymistä. Esimerkkejä tyylittelyihin liittyvistä tehtävistä olisi esimerkiksi päävalikossa olevien alakategorioiden sisältöjen yksilöinti. Alakategorioiden näkymissä on käytössä suurelta osin projekteissa Akeneossa oletusväritys sekä -kuvakkeet. Aloitusnäkymän yksilöintiä voitaisiin myös kehittää esimerkiksi erilaisilla asiakkaan verkkokaupassa käyttämällä värityksellä tai kuvakkeilla tulevissa projekteissa.

Edistyneempiä kehittämistehtäviä Akeneossa olisivat esimerkiksi päävalikon muokkaaminen asiakkaan toivomaksi. Päävalikon muokkaamista voisi olla esimerkiksi erilaisten näkymien lisäämistä päävalikkoon suoraan alakategoriaksi. Näiden lisäksi Akeneossa on lukuisia tuotehallinnan toiminnallisuuksiin ja eri verkkokaupan integraatioihin liittyviä tehtäviä, joiden muokkaamiseksi tarvitaan syvällisempää Akeneon sekä integraatioiden toimintaperiaatteiden tuntemusta. Edistyneemmät kehittämistehtävät toteutettaisiin asiakasprojektien kehitysympäristössä, jolloin asiakkaan omat yksilölliset toiveet tehtävään liittyen Akeneossa tulisi huomioida suunnitteluvaiheesta alkaen tarkemmin.

6 POHDINTA

Kokonaisuutena Akeneon kehittämistyössä käyttöliittymän yksilöimisen valmistumiseen kului ajallisesti ennakoitua kauemmin kuin alkuperäisen suunnitelman perusteella oli varattu aikaa. Tähän vaikuttivat Akeneon kehittämiseen vaadittavan tietotaidon kasvattaminen sekä Pinjan organisaatiossa tapahtuneet sisäiset muutokset. Kehittämistehtävän haastavuudesta huolimatta työ opetti teknisen dokumentaation tiedonhakua sekä Akeneo-ohjelmoimisen perusteiden hallintaa.

Akeneon paikalliseen kehittämisympäristöön asentamisessa suurimmat haasteet tuotti kehittämistehtävän alkuvaiheiden ohjemateriaalin saatavuus. Ohjemateriaalin niukkuus sekä hajanaisuus johti esimerkiksi yhteensopivuusongelmiin uusimman version 6.0 käytössä. Omat haasteensa tuotti myös Linuxin käyttöjärjestelmässä erityisesti pääkäyttäjänä toteutettujen komentojen (sudo) erottaminen tavallisena käyttäjänä toteutetuista komennoista. Suurin osa ilmaiseksi saatavilla olevasta Akeneon materiaalista on suunnattu loppukäyttäjälle, jolloin yleiskuvan saaminen asentamisen eri vaiheista tuottaa aloittelevalle kehittäjälle haasteita. Akeneon kehittäjille suunnattua teknistä opetusmateriaalia on myös osittain maksumuurin takana. Saatavilla olevasta teknisistä ohjemateriaaleista on kuukausien aikana saanut, usean uudelleenasentamisen, jälkeen käsityksen tarvittavista toimenpiteistä asentamisen prosessissa.

Kehittämistehtävänä Akeneon käyttöliittymän yksilöiminen on ollut kasvattava kokemus, jossa onnistuminen on vahvistanut omaa ammattitaitoa sekä tietoisuutta oman osaamisen kehittämisen tarpeesta Akeneon osalta. Erityisesti kehittäjiltä saadut neuvot sekä tuki ovat olleet tärkeässä asemassa oman ammattitaidon kehittämisessä. Tulevissa kehittämistehtävissä onnistuminen on jatkossa helpompaa ja nopeampaa Akeneon perusteiden hallitsemisen myötä.

LÄHTEET

Abraham, J. 2014. Product Information Management. Springer International Publishing. Cham. Luettu: 7.12.2022. https://www.sci-hub.ru/10.1007/978-3-319-04885-7.

Adobe 2022. Find a Partner. Luettu: 16.12.2022.https://solutionpartners.adobe.com/s/directory/.

Akeneo 2019. Akeneo Named a Leader in IDC MarketScape on PIM. Luettu: 29.3.2022. https://www.prnewswire.com/news-releases/akeneo-named-a-leader-in-idc-marketscape-on-pim-300975223.html.

Akeneo 2022a. What is a PIM? Luettu: 2.2.2022. https://www.akeneo.com/what-is-a-pim/.

Akeneo 2018. Akeneo Recognized as Magento Premier Technology Partner. Luettu: 31.3.2022. https://www.epicos.com/article/331267/akeneo-recognized-magento-premier-technology-partner..

Akeneo 2022b. About Akeneo. Luettu: 2.2.2022. https://www.akeneo.com/about-us/.

Akeneo 2022c. Install Akeneo PIM for development with Docker — Akeneo PIM documentation. Luettu: 19.10.2022. https://docs.akeneo.com/5.0/install_pim/docker/installation_docker.html.

Akeneo 2022d. How to contribute to the frontend part of the application — Akeneo PIM documentation. Luettu: 17.10.2022. https://docs.akeneo.com/5.0/contribute_to_pim/front.html.

Akeneo PIM 2022. Luettu: 4.2.2022. https://pinja.com/en/services/trade-service-businesses/product-information-management.

Battistello, L. 2020. The digitalization journey of product information management. Technical University of Denmark. Luettu: 27.1.2022. https://orbit.dtu.dk/en/publications/the-digitalization-journey-of-product-information-management.

Chacon, S. & Straub, B. 2014. Pro git. Luettu: 28.11.2022. https://git-scm.com/book/en/v2.

Chaudhary, M. & Kumar, A. 2014. PhpStorm cookbook: discover over 80 recipes to learn how to build and test PHP applications efficiently using the PhpStorm IDE. Packt Publishing. Birmingham, England.

Chen, I.J. 2001. Planning for ERP systems: analysis and future trend. Business Process Management Journal, 7, 5, s. 374–386. Luettu: 11.3.2022. https://www.emerald.com/insight/content/doi/10.1108/14637150110406768/full/html.

Combe, T., Martin, A. & Di Pietro, R. 2016. To Docker or Not to Docker: A Security Perspective. IEEE Cloud Computing, 3, 5, s. 54–62. Luettu: 11.3.2022. https://sci-hub.se/10.1109/mcc.2016.100.

Eine Bastien, Jurisch, M. & Quint, W. 2017. Systems | Free Full-Text | Ontology-Based Big Data Management | HTML. Luettu 27.1.2022.https://www.mdpi.com/2079-8954/5/3/45/htm.

Explore Docker's Container Image Repository | Docker Hub 2022. Luettu: 27.1.2022. https://hub.docker.com/search?offering=community&operating_system=linux&q=&type=edition.

Getting started | Less.js 2022. Luettu: 17.12.2022. https://lesscss.org/.

Grizaut, C. 2018. Everything you need to know about PIM. Akeneo. Luettu: 2.2.2022. https://magento.com/sites/default/files8/2019-01/product-information-management-101.pdf.

Jayanandana, N. 2018. Build a Docker Image just like how you would configure a VM. Luettu: 17.11.2022. https://medium.com/platformer-blog/practical-guide-on-writing-a-dockerfile-for-your-application-89376f88b3b5.

Laaziri, M., Benmoussa, K., Khoulji, S., Larbi, K. & Yamami, A. 2019. A comparative study of laravel and symfony PHP frameworks. International Journal of Electrical and Computer Engineering, 9, s. 704–712. Luettu: 8.2.2022. https://research.amanote.com/publication/EZDa1HMBKQvf0Bhi3IQe/a-comparative-study-of-laravel-and-symfony-php-frameworks.

Lamont, J. 2016. E-commerce: managing complexity. KM World, 25, 9, s. 8–10. Luettu: 25.1.2022. https://web-p-ebscohost-com.ezp.oamk.fi:2047/ehost/detail/detail?vid=0&sid=c2b78102-d79d-4a62-83e2-1c00dd765ef7%40redis&bdata=JnN-pdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=119439712&db=buh.

Lantz, A. & Agné, F. 2014. Vägen till en användarvänlig webbapplikation. s. 64.

MacIntyre, P. 2010. PHP: The Good Parts: Delivering the Best of PHP. O'Reilly Media, Inc.

Matos, A.I. de S. 2021. Product information management for complex modular security systems. Luettu: 16.3.2022. https://comum.rcaap.pt/handle/10400.26/39100.

MVC Framework Introduction 2022. Luettu: 11.12.2022. https://www.geeksforgeeks.org/mvc-framework-introduction/.

Nickoloff, J. & Kuenzli, S. 2019. Docker in Action, Second Edition. Simon and Schuster.

Pinja 2022. Business sectors. Luettu: 1.12.2022. https://pinja.com/en/business-sectors.

Pop, D.-P. & Altar, A. 2014. Designing an MVC Model for Rapid Web Application Development. Procedia Engineering, 69, s. 1172–1179. Luettu: 31.3.2022. https://doi.org/10.1016/j.proeng.2014.03.106.

Prokofyeva, N. & Boltunova, V. 2017. Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. Procedia Computer Science, 104, s. 51–56. Luettu: 11.4.2022. https://dl.acm.org/doi/10.1016/j.procs.2017.01.059.

Richter, J. 2022. Open Source PIM Systems. Luettu: 7.12.2022. https://parsionate.com/en/resources/magazine/open-source-pim-systems/.

Roehl-Anderson, J.M. 2010. IT Best Practices for Financial Managers. John Wiley & Sons.

Salehi, S. 2016. Mastering Symfony. Packt Publishing Ltd.

SAS, I. 2022. What is data mining? Luettu: 14.3.2022. https://www.sas.com/fi_fi/insights/analytics/data-mining.html.

Sparks, B.A., Perkins, H.E. & Buckley, R. 2013. Online travel reviews as persuasive communication: The effects of content type, source, and certification logos on consumer behavior | Elsevier Enhanced Reader. Luettu: 10.12.2022. https://sci-hub.se/10.1016/j.tourman.2013.03.007.

Symfony - Architecture 2022. Luettu: 8.12.2022. https://www.tutorialspoint.com/symfony/symfony_architecture.htm.

Tapalinen, S. 2022. Akeneo ohjelmistokehittäjän kokemukset Akeneon käytettävyydestä.

Turnbull, J. 2014. The Docker Book: Containerization Is the New Virtualization. James Turnbull. Luettu: 28.11.2022. https://books.google.fi/books?hl=en&lr=&id=4xQKBAAAQBAJ&oi=fnd&pg=PA1&dq=docker+virtualization&ots=wx8Mdw4hHT&sig=4H-jI1k8_nMYBnGh6BI9T1AaOV0&redir_esc=y#v=onepage&q=docker%20virtualization&f=false.

Varanka, J. 2022. Magento 2 - benefits and features. Luettu: 5.12.2022. https://blog.pinja.com/en/magento-2-benefits-and-features.

Waeosri, W. 2022. How to deploy and run Real-Time Java Application with Maven in Docker. Luettu: 30.8.2022. https://developers.refinitiv.com/en/article-catalog/article/how-to-deploy-and-run-real-time-java-application-with-maven-in-d.

Zaninotto, F. & Potencier, F. 2007. The Definitive Guide to symfony. Apress.