Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Responzivní informační systém pro malou firmu

Diplomová práce

Autor: Bc. Lukáš Senohrábek

Studijní obor: AI2

Vedoucí práce: doc. Ing. Filip Malý, Ph.D.

Hradec Králové duben 2015

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedených zdrojů.

V Hradci Králové dne Lukáš Senohrábek

Poděkování

Rád bych tímto poděkoval svému vedoucímu práce, doc. Ing. Filipu Malému, Ph.D., za podnětné rady, které mi při psaní práce poskytl.

Velké díky také patří mým rodičům a blízkým, kteří mě během studií neustále podporovali.

Anotace

Tématem diplomové práce je návrh a implementace responzivního informačního systému pro malou firmu. Mezi hlavní funkce systému patří snadné použití na PC i chytrém telefonu, správa 3D modelů, zakládání a evidence objednávek včetně zákazníků nebo kalendář pro plánování úkolů a schůzek. Práce se věnuje teoretickým základům informačních systémů, analýze firmy a jejích potřeb, návrhu a vlastní implementaci systému.

Systém je postaven na PHP frameworku Yii ve verzi 2, který je, společně s dalšími použitými technologiemi, popsán v praktické části práce. Současně byly v práci popsány jednotlivé kroky návrhu a také některé praktické úkoly, na které se při implementaci narazilo.

V závěru jsou shrnuty dosažené výsledky a nastíněny možnosti budoucího vývoje.

Annotation

Title: Responsive information system for small business

Theme of diploma thesis is design and implementation of responsive information system for small business. Main features of the system include ease of use for PC and smartphones, 3D models management, establishment and registration of orders and customers or calendar for scheduling tasks and appointments. This work focuses on theoretical foundations of information systems, analysis of the company and its needs, and design and implementation of the system.

The system is bulit on PHP framework Yii version 2, which is, with other used technologies, described in practical part. Next, this work describes various steps in design and some practical tasks, that were encountered during implementation.

Conclusion summarizes achievements and possibilities of future development.

Obsah

1 Úvod

Každá firma, ať již začínající, nebo na trhu dlouho zavedená, vyžaduje k dobrému fungování kvalitní, tedy přesné a včasné informace. Tyto informace ovlivňují veškeré procesy, které ve firmě probíhají. Ovšem nestačí tyto informace pouze mít, je také nutné je umět zpracovat a porozumět jim. Až tehdy představují pro firmu konkurenční výhodu. Pokud má firma dostatek informací a umí je správně interpretovat, může díky nim růst, vyhledávat nové příležitosti nebo snižovat náklady na provoz. To vše jí zajistí kvalitní informační systém. Bez takového systému může docházet ke špatné komunikaci a z ní vyplynout problémy, které mohou způsobit časové, anebo finanční problémy.

Opravdu dobrý informační systém není levná záležitost. Jeho implementace a zaškolení zaměstnanců, kteří s ním budou pracovat, je nákladný proces z časového i finančního hlediska. Kvalitní informační systém bývá v nejlepším případě vyvíjený přímo pro potřeby konkrétní firmy. Jen tak dokáže dokonale reflektovat vnitřní procesy a zjednodušit práci všem zúčastněným pracovníkům.

Tato diplomová práce se právě na takový individuální informační systém a proces jeho návrhu a implementace zaměřuje.

2 Cíl a metodika práce

Cílem této práce je zanalyzovat potřeby mladé firmy, která se zabývá zakázkovou výrobou plastových výrobků na 3D tiskárnách, a na základě této analýzy pak navrhnout co možná nejlepší řešení a nakonec kompletně implementovat vlastní informační systém. Součástí řešení bude i návrh, stavba a konfigurace serveru, na kterém nově implementovaný systém poběží. Protože se jedná o velmi mladou firmu, jejíž fungování není stále ještě položeno na pevně daných firemních procesech, budou současně s probíhající analýzou a konzultacemi tvořeny i tyto procesy.

V diplomové práci budou využity některé metody pro analýzu současného stavu ve firmě, jako je například SWOT nebo PEST.

Nejdůležitější firemní procesy budou popsány nejen slovně, ale i graficky. Požadavky pracovníků na informační systém budou podrobeny analýze a na základě výsledků bude navržen a následně implementován odpovídající informační systém.

3 Základní teorie

Tato část diplomové práce obsahuje shrnutí nejdůležitějších pojmů z oblasti informatiky a systémů. Na začátku se zaměřuje na obecný popis a přejde k analýze a návrhu. Sepsanými poznatky se pak řídí celý proces návrhu a implementace IS v praktické části.

3.1 Nejdůležitější pojmy

Mezi základní pojmy v procesu vývoje informačního systému patří informatika, systém a informační systém. Všechny tyto pojmy jsou v kapitole popsány a vysvětleny.

3.1.1 Informatika a informace

Informatika je multidisciplinární obor, jehož předmětem je tvorba a užití informačních systémů v organizacích a společenstvích, a to na bázi moderních informačních technologií. Zjednodušeně by se dalo říci, že informatika je věda, která se zabývá zpracováním informací (např. vyhledávání, třídění, přenášení a/nebo ukládání). [1]

Informace je chápána jako nějaký údaj o prostředí, jeho stavu a procesech, které v něm probíhají. Informacemi rozumíme data, kterým jejich uživatel přisuzuje určitý význam a které uspokojí konkrétní objektivní informační potřebu svého příjemce. Informace tak snižuje nebo dokonce odstraňuje neurčitost systému. Informace v podobě dat může být textová (dokument), grafická (obrázek, video) nebo zvuková (nahrávka), data jsou tedy nositelem informace. Na rozdíl od dat má informace nehmotný charakter, nemůžeme ji tedy skladovat, ale vždy je spojena s nějakým fyzickým pochodem. [1]

3.1.2 Systém

Teorie systémů chápe systém jako uspořádanou rozmanitost materiálních nebo ideálních objektů. Je jím množina prvků a vztahů mezi nimi, jež může být považována za jeden celek a vykazuje určité vlastnosti. Jinak řečeno, systém je množina vzájemně propojených prvků, které musí pracovat dohromady pro celý systém tak, aby tento systém naplnil daný účel (daný cíl). Tzn., že i když každý jednotlivý prvek systému je dobře navržen a pracuje efektivně, systém neplní svoji funkci, pokud tyto prvky nepracují dohromady. [1] [2]

3.1.3 Informační systém

Informační systém můžeme chápat jako určitý typ komunikačního média, jehož cílem je odstranit bariéry v přístupu k informacím. Činí tak pomocí zpracování informací, které

do systému vstupují a transformuje je na informace ze systému vystupující – časově, prostorově i formou tak, aby byly využitelné lépe, než v původním stavu. Aby mohl zpracovávat data, potřebuje ke své činnosti určité nástroje, metody a znalosti, které se nazývají informační technologie. Informačními technologiemi rozumíme hardwarové a softwarové prostředky pro získávání, zpracování, uchování, přenos a distribuci dat. [1] [2]

Do celkové funkce informačního systému se také promítá další nezanedbatelná položka – okolí. Tím jsou myšleny veškeré objekty, které ovlivňují systém změnou svých vlastností, a objekty, které naopak mění své vlastnosti v závislosti na stavu systému.

3.2 Typy informačních systémů

Informační systémy můžeme dělit do různých kategorií podle několika hledisek, například účel, velikost, počet uživatelů nebo rozsah použití.

3.2.1 Informační systémy organizací

Informace jako ekonomický zdroj – podnikový informační systém, který je provozovaný v rámci konkrétní organizace. Jeho účelem je správa informací a znalostí a jejich integrace do podnikových procesů za podpory informačních a komunikačních technologií. Obsažené informace jsou chápány jako jeden z ekonomických zdrojů organizace. Rozlišujeme systémy podporující vlastní činnosti a služby organizace a takzvané manažerské systémy, které podporují řídící a administrativní funkce. Mezi podnikové informační systémy řadíme ERP, MIS, ESI, BI, APS, SCM, CRM.

Dále mohou tyto informační systémy podporovat účel, kvůli kterému organizace existuje. [2]

3.2.2 Veřejné informační systémy

Informační systém jako produkční systém organizace. Základním produktem nebo službou takové organizace jsou informace – například tisk, knihovny, zpravodajské agentury nebo informační instituce. I v tomto případě ovšem musí mít tato organizace vlastní informační systém na podporu vlastního řízení. [2]

3.2.3 Státní informační systém

Informační systémy státní správy a samosprávy, informační systémy veřejné správy (GIS = government information system). Jedná se o specifický typ informačního systému

organizace. Jeho účelem je podporovat činnosti při výkonu veřejné správy a poskytovat veřejné informační služby včetně informací o subjektech veřejné správy. Nejdůležitější součást datové základny tvoří evidence (registry) základních skutečností nezbytných pro výkon veřejné správy, např. evidence obyvatel nebo ekonomických subjektů. [2]

3.3 ERP systém

Původně byly ERP systémy (Enterprise Resource Planing, systémy pro podporu plánování) určeny pro průmyslové podniky jako nástroj pro řízení výroby, ale v dnešní době nacházejí uplatnění i v jiných oblastech, jako jsou lidské zdroje, finance, logistika. Proto je v některých případech lepší mluvit o celopodnikových informačních systémech. [3]

„*Informační systém kategorie ERP definujeme jako účinný nástroj, který je schopen pokrýt plánování a řízení hlavních interních podnikových procesů (zdrojů a jejich transformaci na výstupy), a to na všech úrovních, od operativní až po strategickou.*“ [4]

Interním podnikovým procesem je myšlen takový proces, nad nímž má management plnou kontrolu a je tedy jeho vlastníkem. Mezi tyto klíčové procesy řadíme výrobu, nákupní, prodejní a výrobní logistiku, lidské zdroje a ekonomiku.

Mezi hlavní požadavky kladené na ERP systémy patří:

- Realizace měřitelných přínosů v oblasti snižování celé struktury nákladů vznikajících neefektivním řízením firmy.
- Realizace neměřitelných přínosů v oblasti řízení podnikových procesů a dostupnosti informací v reálném čase.

ERP systémy definuje pět základních vlastností:

- Automatizace a integrace hlavních podnikových procesů
- Sdílení dat, postupů a jejich standardizace přes celý podnik
- Vytváření a zpřístupňování informací v reálném čase
- Schopnost zpracovávat historická data
- Celostní přístup k prosazování ERP koncepce

K hlavním požadovaným vlastnostem, které přímo souvisejí s technologickými aspekty ERP systému, patří výkonnost, spolehlivost a bezpečnost. Podmínkou, kterou je nutné

pro zajištění těchto vlastností splnit, je plnohodnotný provoz ERP systému na architektuře klient/server. Výkonnost a spolehlivost systému dále závisí na využití odpovídajících hardwarových a softwarových komponentů a poskytnutí požadované kapacity systémových prostředků. Další základní požadavky, jako zabezpečení komunikace mezi serverem a klientem, sledování historie jednotlivých záznamů, autentifikace uživatelů a definice přístupových práv (a mnoho dalších), pak zajišťují bezpečnost ERP systému. [4]

3.3.1 Problémy českých podniků při realizaci ERP projektů

Realizace ERP řešení není zcela jednoduchou záležitostí, o čemž svědčí řada ukončených projektů nebo projektů, které nesplnily očekávání klientů. Z výzkumu provedeného v letech 2006 a 2010 bylo sestaveno deset kritických faktorů, které byly během výzkumu nejčastěji zmíněny. Tyto faktory jsou prezentovány jako základní nedostatky ERP projektů: [3] [4]

- Absence podnikové a IT strategie, dle nichž by se měl projekt realizovat
- Neschopnost správně formulovat zadání ERP projektu
- Nedostatek kvalifikovaných pracovníků pro realizaci ERP projektu
- Snaha o dosažení co nejnižší ceny ERP projektu na úkor kvality řešení
- Neschopnost věcné komunikace s dodavatelem i uvnitř podniku
- Snaha přesunout veškerou odpovědnost za projekt na dodavatele
- Definování implementačního týmu a rozdělení kompetencí
- Technická připravenost na řešení projektu a špatná kvalita dat
- Nedostatek času na realizaci ERP projektu
- Snaha ušetřit za konzultační služby a školení

Z výzkumu lze nadále konstatovat, že čtyři z nejčastěji zmíněných faktorů spojuje neznalost a nepochopení důležitosti strategického řízení a významu informačního systému pro růst výkonnosti a hodnoty podniku. Nelze také opomenout, že jednou z hlavních příčin je přetrvávající informační a znalostní bariéra a podceňování úlohy lidí, ať již zúčastněných v procesu implementace, nebo jako koncových uživatelů. [4]

3.4 Metody pro analýzu

Analýza slouží ke zkoumání složitějších skutečností na jednodušší. Tato kapitola popisuje tři metody, které slouží k analýze a zhodnocení stavu podniku z vnitřních a vnějších hledisek.

3.4.1 SWOT

SWOT je typ strategické analýzy, která nahlíží na podniky či organizace ze čtyř hledisek. Těmi jsou silné stránky (strengths), slabé stránky (weaknesses), příležitosti (opportunities) a ohrožení (threats). Tato analýza poskytuje podklady pro formulaci rozvojových směrů a aktivit, podnikových strategií a strategických cílů. Podstatou SWOT analýzy je tedy identifikovat klíčové silné a slabé stránky podniku a klíčové příležitosti a hrozby vnějšího prostředí podniku. [5]

SWOT analýza hodnotí vnitřní a vnější faktory. Vnitřní faktory zahrnují hodnocení silných a slabých stránek. Příkladem vnitřních faktorů je výkonnost a motivace pracovníků, efektivita procesů, logistické systémy a podobně. Silné a slabé stránky podniku jsou ty faktory, které vytvářejí nebo naopak snižují vnitřní hodnotu firmy (aktiva, dovednosti, podnikové zdroje atd.). Nejčastějšími vstupy jsou finanční analýzy, hodnocení pomocí EFQM[1], analýza hodnotového řetězce a další. [5]

Vnějšími faktory zahrnují hodnocení příležitostí a hrozeb, které souvisí s okolním prostředím organizace. Tyto faktory nemůže podnik tak dobře kontrolovat. Nicméně i přes to je může alespoň identifikovat pomocí vhodné analýzy konkurence, demografických, ekonomických, politických, sociálních a dalších faktorů působících v okolí podniku. V běžné praxi tvoří SWOT analýzu soubor potřebných externích i interních analýz podniku. Mezi nejčastější vstupy tak patří sektorová analýza nebo analýza konkurenčního postavení. [5]

3.4.2 PEST

PEST analýza je další z důležitých nástrojů strategického řízení, který hodnotí pouze vnější faktory podniku, které nemůže svými aktivitami, nebo jen velmi obtížně, zvládnout. PEST je zkratkou pro analýzu politických, ekonomických, sociálních a technologických faktorů, která přichází obvykle v době, kdy se firma rozhoduje

[1] EFQM Excellence Model je model vyvinutý jako rámec pro uplatňování metod řízení jakosti v organizaci.

o nějakém velkém kroku (například vstup na nové trhy, velký projekt atp.). Důvodem pro PEST analýzu však mohou být třeba i investice do podniku.

V současné době je možné se setkat s různými rozšířeními, například PESTLE, kdy bylo do analýzy přidáno právní (legal) a životní (environmental) prostředí. [6]

3.4.2.1 Politické prostředí

Politickými faktory není myšlena problematika politických stran, ale spíše stabilita politické scény a legislativního rámce. Kromě stability je důležitý také obsah, sledují se tak všechny zákony a návrhy, které ovlivňují oblast působení firmy, a také chování regulačních orgánů (energetika, telekomunikace a podobně).

3.4.2.2 Ekonomické prostředí

Ekonomika je důležitá především pro odhad ceny pracovní síly i pro odhad cen produktů a služeb. V této části analýzy se sledují otázky daní (DPH, spotřební daň) a cel, stabilita měny, úrokové sazby, makroekonomické ukazatele (např. HDP), ale patří sem také různé pobídky investorů nebo podpora exportu.

3.4.2.3 Sociální prostředí

Tato oblast analýzy je důležitá především pro firmy podnikající v oblasti prodeje koncovým zákazníkům. Řeší demografické údaje, etnické a náboženské otázky, oblast médií a jejich vlivu, události jako veletrhy nebo konference i trendy životního stylu. Analýza sociálního prostředí je výrazně zjednodušena faktem, že z velké části je již zpracována národním statistickým úřadem.

3.4.2.4 Technologické prostředí

Analýza technického prostředí se zabývá otázkami infrastruktury (všechny druhy dopravy, včetně surovin nebo energií), stavem rozvoje a zaměření průmyslu a stavem vědy, výzkumu a podpory vědy a vysokého školství. Někdy může být do této části vyčleněna také část legislativy, která se zabývá regulací průmyslu. Tu mají mnohdy na starosti nižší právní formy jako například prováděcí vyhlášky.

3.4.3 HOS 8

Metoda vyvíjená na Ústavu informatiky Podnikatelské fakulty VUT použitelná ve fázi přípravy informační strategie. Základní filozofie metody HOS spočívá v ohodnocení úrovně jednotlivých součástí informačního systému a nalezení těch složek, které

negativně ovlivňují celkovou úroveň systému. Cílem metody HOS je posouzení klíčových oblastí informačního systému firmy a zjistit, zda všechny tyto oblasti jsou na stejné, či blízké úrovni. [7]

Oblasti hodnocení informačního systému metodou HOS:

- **Hardware** - V této oblasti je zkoumáno technické vybavení firmy, hardware.
- **Software** - Tato oblast zahrnuje zkoumání programového vybavení, jeho funkcí, snadnosti používání a ovládání.
- **Orgware** - Tato oblast zahrnuje pravidla pro provoz informačních systémů, doporučené pracovní postupy, bezpečnostní pravidla.
- **Peopleware** – Oblast zahrnuje zkoumání uživatelů informačního systému. Zaměřuje se především na pracovníky z pohledu jejich povinností vůči informačnímu systému.
- **Dataware** – Oblast zkoumá data ve vztahu k jejich dostupnosti, správě, bezpečnosti a potřebě užití v procesech podniku.
- **Customers** (zákazníci) – Zákazník informačního systému může být chápán jako skutečný zákazník, například uživatel internetového obchodu, nebo jako kterýkoliv pracovník podniku, který potřebuje systém a jeho výstupy ke své práci.
- **Suppliers** (dodavatelé) – Dodavatelem je míněn ten, kdo zajišťuje provoz informačního systému. Může se jednat o externí firmu nebo o pracovníky, kteří v podniku zajišťují podporu informačního systému.
- **Management** – Tato oblast zkoumá řízení informačních systémů ve vztahu k informační strategii, důslednosti uplatňování stanovených pravidel a vnímání koncových uživatelů informačního systému.

3.5 Modelování procesů

Modelování procesů pomáhá pochopit strukturu vnitřních procesů podniku. Pomocí modelování procesů je možné popsat procesy, podprocesy a činnosti. Vzniklé modely mohou sloužit jako podklad pro tvorbu informačního systému nebo pro jejich optimalizaci.

3.5.1 EPC

Metoda EPC (z anglického Event-driven Process Chain) patří k jedné z nejrozšířenějších. Metoda, jak plyne i z jejího názvu, spočívá v řetězení událostí a aktivit do posloupností k dosažení požadovaného cíle. EPC nám tak pomáhá určit, jak by se měl vlastně takový proces chovat, jak bude realizován a především jaký bude mít průběh. [8]

Na EPC modelu se můžeme setkat s těmito prvky:

- **Aktivita** (activity) – Na diagramu reprezentuje nějakou činnost a je znázorněna obdélníkem se zakulacenými rohy.
- **Událost** (event) – Znázorňují situace, které mohou nastat před nebo po vykonání aktivity. Na diagramu se obvykle střídá s aktivitou, z toho lze usoudit, že událost je obvykle vstupní podmínkou některé aktivity a zároveň výstupní podmínkou aktivity jiné. Diagram vždy začíná počáteční a končí ukončující aktivitou. Jsou znázorněny šestiúhelníkem.
- **Logická spojka** (connection) – Prvky, které nám umožňují větvit tok procesu, případně ho opět spojovat. V EPC existují tři typy spojek, které se liší počtem požadovaných splnění podmínek – AND (všechny), OR (alespoň jedna) a XOR (právě jedna). V diagramu je spojka vyznačena jako kolečko s příslušným znaménkem.
- **Organizační jednotka** (organization unit) – Určuje osobu nebo oddělení, které je zodpovědné za danou aktivitu. Na diagramu je vyobrazena jako elipsa s pruhem.
- **Informace, materiál, zdroj** (information, material, resource) – Element, který může znázorňovat například objekty nebo informace, které jsou nutné pro provedení aktivity nebo jsou v průběhu aktivity spotřebovány. Na diagramu jsou znázorněny obdélníkem.
- **Kontrolní tok** (control flow) – Pomocí kontrolních toků jsou všechny ostatní elementy propojeny, umožňují tak zobrazit časovou posloupnost aktivit. Celkem logicky jsou tedy znázorněny šipkami.

3.5.2 E-R diagram

Entity - Relationship Diagram, česky diagram vztahů mezi entitami, je jednou z nejoblíbenějších technik pro vyjádření návrhu dat pro datově orientované systémy. ERD nám dovoluje graficky analyzovat a zobrazit datový model systému pomocí objektů, které nás zajímají, jejich vlastností a vzájemných vztahů. [2]

Základní prvky a symboly ERD:

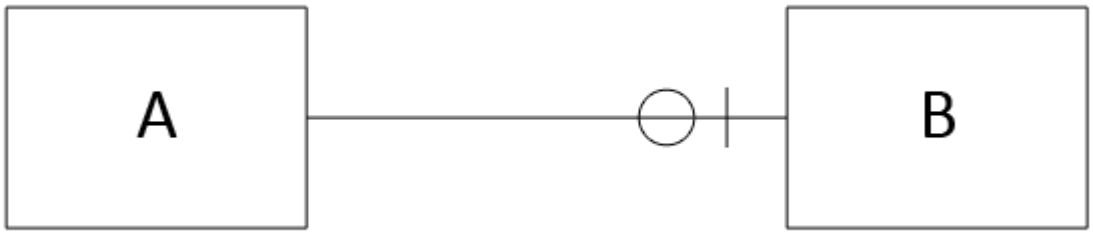
- **Entita** – symbol obdélníku s názvem (podstatné jméno, např. materiál). Rozlišitelný a identifikovatelný objekt světa objektů, o kterých sbíráme data.
- **Vztah** – symbol kosočtverce včetně čar, které spojují entity, který určuje vztah mezi entitami. Jako název uvádíme sloveso (nakupuje).
- **Atributy** – symbol kružnice s názvem (podstatné jméno, např. cena). Přiřazuje entitám nebo vztahům hodnotu, která určuje nějakou podstatnou vlastnost.
- **Integritní omezení** – definuje vlastnosti entity, vztahu nebo atributu, např. datový typ nebo kardinalita vztahu.

3.5.2.1 Kardinalita a členství ve vztahu

Kardinalita určuje počet prvků nějakého vztahu, neboli kolik výskytů (instancí) jedné entity má vztah k výskytu druhé entity, zatímco členství možnost (ne)existence výskytu partnerské entity – vyžaduje výskyt jedné entity výskyt entity druhé?

Kardinalitu zobrazujeme pomocí písmen a číslic, číslic a symbolů nebo „vraní stopy“. Na obrázcích je příklad použití vraní stopy.

Žádný nebo jeden – 0 : 1, ke každé entitě A se váže žádná nebo jedna entita B



Obrázek 1: Kardinalita žádný nebo jeden, zdroj: vlastní

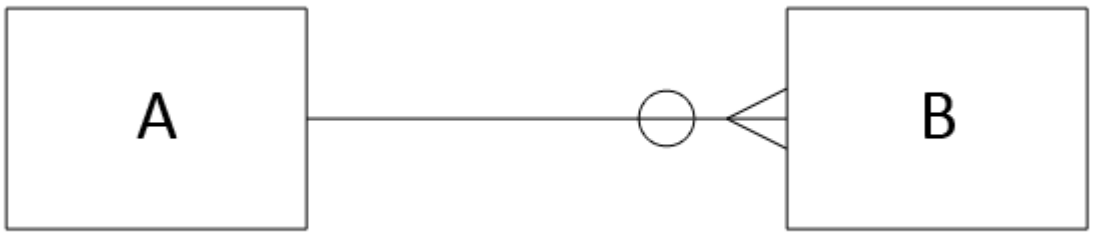
Právě jeden – 1 : 1, každá entita A se váže právě k jedné entitě B

Obrázek 2: Kardinalita právě jeden, zdroj: vlastní

Jeden nebo více – 1 : N, ke každé entitě A se váže jedna nebo více entit B

Obrázek 3: Kardinalita jeden nebo více, zdroj: vlastní

Žádný, jeden nebo více – 0 : N, k entitě A se váže žádná, jedna nebo více entit B



Obrázek 4: Kardinalita žádný, jeden nebo více, zdroj: vlastní

Více než jeden – M : N, ke každé entitě A se váže více než jedna entita B

Obrázek 5: Kardinalita více než jeden, zdroj: vlastní

3.6 Životní cyklus IS

SLC, neboli Systems Life Cycle (životní cyklus systému) je časové období životnosti systému od jeho zadání až k jeho zániku, resp. jeho nahrazení jiným systémem. Označuje se tak etapizace života projektu a speciální metodika pro postup práce na projektu (je to jedna ze speciálních systémových metodologií). Další metodikou je například MDIS[2].

3.6.1 Popis klíčových etap životního cyklu IS

Životní cyklus informačního systému se skládá z jednotlivých etap, které slouží jako stavební kameny při jeho budování.

3.6.1.1 Předanalytická fáze

Cílem této fáze je sestavit základní rámec požadavků, cílů a funkcí. Základem celkového návrhu, vývoje i úprav stávajícího systému jsou požadavky uživatelů – základní podmínky, které musí systém splňovat. V této části se musí všechny požadavky shromáždit, rozebrat a odhadnout dobu realizace a náklady.

3.6.1.2 Analýza systému

V této fázi probíhá modelování budoucího systému na konceptuální úrovni. Veškeré chyby ve struktuře dat i systému, které se zde neodhalí, jsou později velice obtížně odstranitelné, proto je tato fáze klíčová.

3.6.1.3 Návrh

Výsledkem předchozí fáze je projektová studie systému, která se modelováním na technické úrovni převádí na detailnější schémata a procesy. Tato studie slouží jako podklad pro realizaci systému a podklad pro podmínky předání a testování.

[2] Multidimensional Development of Information System, publikována na VŠE v roce 1992

3.6.1.4 Implementace

Fáze programování vlastního informačního systému, kterého se účastní jak programátoři, tak i analytik nesoucí zodpovědnost za správnost řešení. Podkladem pro tuto fázi jsou veškeré dokumenty, které vznikly v průběhu předchozích etap. Zdroj [2] uvádí v rámci této etapy také testování a vlastní zavádění systému do provozu. Během testování se provádí připravené testy na hotovém informačním systému, kdy je potřeba vyzkoušet možné reakce systému na zadávaná data a případné nedostatky opravit. Zavádění systému do provozu je popsáno v kapitole 3.7.

3.6.1.5 Zkušební provoz, rutinní provoz a údržba systému

Během zkušebního provozu je poskytovatel povinen zajistit okamžitý servis a také odstranit chyby zjištěné během provozu, případně dořešit dodatečné požadavky uživatelů, ale pouze v rámci původního návrhu.

Rutinní provoz a údržba jsou už finální fází projektu, během které je systém užíván koncovými uživateli. Údržbou systému je zajištěn správný provoz systému, úprava parametrů aplikací nebo změny některých programů, aby splňovaly nové požadavky uživatelů. Kromě zajištění optimálního provozu, zabezpečení a ochrany dat patří v neposlední řadě do této etapy i opětovné školení uživatelů. [2] [9]

3.7 Strategie zavádění IS do firmy

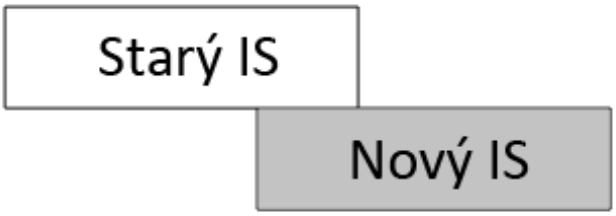
Cílem je zavést informační systém do rutinního provozu podniku. Tato etapa se může zdát jako nejméně obsáhlá, ale opak je pravdou, protože tento procesu zahrnuje konverze databází, přípravu uživatelů a další organizační opatření jako instalaci technického a programového vybavení, školení uživatelů, testy a další. Je potřeba zajistit, aby přechod podniku na nový systém neomezoval běžný pracovní režim uživatelů (zaměstnanců) a je také nutné dát uživatelům čas, aby si na nový systém zvykli. [10]

Postupů pro zavádění informačního systému do rutinního provozu je veliké množství. Liší se od sebe rychlostí, zaváděcí metodou apod. Dle [11] patří mezi používané strategie:

3.7.1 Souběžné zavádění

Informační systém je zaveden souběžně na všech pracovištích najednou. Tento postup je vhodné použít při zavádění jednodušších informačních systémů, které nevyžadují

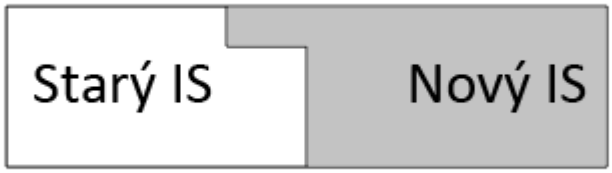
náběhovou fázi zavádění (složitá školení, konverzi dat z předchozích informačních systémů).



Obrázek 6: Souběžné zavádění informačního systému, zdroj: vlastní

3.7.2 Pilotní zavádění

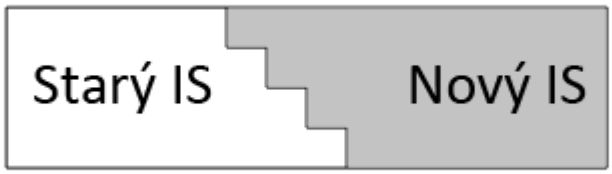
Informační systém se zavede na jednom pracovišti, které je na tuto činnost připraveno. Po zavedení probíhá ověřovací provoz a posléze zde probíhá zacvičování pracovníků ostatních pracovišť. Tento způsob je vhodný pro zavádění kvalitativně odlišných informačních systémů, které vyžadují rozsáhlé testování nového systému v provozních podmínkách. Toto pilotní zavádění umožňuje postupnou transformaci dat z předchozích informačních systémů. V závěru pilotní fáze dochází k zavádění IS na ostatní pracoviště, které jsou již připravena.



Obrázek 7: Pilotní zavádění informačního systému, zdroj: vlastní

3.7.3 Postupné zavádění

Zavádění IS na jednotlivá pracoviště probíhá postupně, bez pilotní fáze. Rychlost zavádění je závislá na připravenosti jednotlivých pracovišť a na složitosti informačního systému. Tento způsob je vhodný pro takový systém, u kterého není nutné provozní ověřování (komerčně dodávaný, převzatý z podobně fungujících pracovišť).



Obrázek 8: Postupné zavádění informačního systému, zdroj: vlastní

3.7.4 Nárazová strategie zavádění

Strategie zavádění, kde najednou ukončíme činnost jednoho informačního systému a po nezbytně nutné pauze spustíme nový. Tento postup je riskantní, používá se tam, kde souběh dvou různých systémů není možný.

Obrázek 9: Nárazová strategie zavádění informačního systému, zdroj: vlastní

V praxi však často nastává nutnost kombinovat jednotlivé postupy. Nejčastější je kombinace postupu nárazového a postupného.

3.8 Responzivní design

Responzivní design je ve světě webdesignu vcelku nový pojem, poprvé ho použil ve svém článku Ethan Marcotte v květnu 2010[3]. Cílem responzivního designu je vytvářet stránky, které zjednoduší prohlížení a zlepší pocit z něho. Pro responzivní design je typická především snadná navigace a čitelnost, ale také minimální nutnost rolování a posouvání při změně velikosti okna prohlížeče. Lze ho tak využít na přístrojích s různou velikostí obrazovky – od osobních počítačů přes tablety až po chytré telefony.

Současně jsou mobilní telefony vhodným výchozím bodem pro práci na responzivním designu – je totiž obvyklé, že se nejprve vytvoří návrh webu pro mobily a následně se vylepšuje s požadavky na responzivní web. [12]

[3] **Marcotte, Ethan**. Responsive Web Design. *A List Apart* [online]. č. 306, 25. 5. 2010. http://alistapart.com/article/responsive-web-design

Přizpůsobení webové stránky probíhá na dvou základních úrovních: [13]

- Proporcionální (flexibilní) design – velikost objektů webové stránky je definována relativně, takže se přizpůsobují svému okolí.
- Media queries – vzhled se přizpůsobuje na základě parametrů zobrazovacího zařízení, typicky velikost průhledu (*viewport* = plocha dostupná pro vykreslení stránky).

3.8.1 Nástroje responzivního webdesignu

Pro vytvoření responzivního designu pro web je nutné využít různé techniky nebo nástroje. Mezi takové nástroje se řadí proporcionální design, media queries, přizpůsobení velikosti displeje a přepočet jednotek podle hustoty pixelů.

3.8.1.1 Proporcionální design

Jazyk CSS definuje dva typy jednotek a to absolutní a relativní. Základním stavebním prvkem responzivního designu jsou objekty definované v relativních velikostech. Relativní jednotky umožňují obecnou definici rozměru objektů, které se přizpůsobí průhledu.

Jednotky relativní k velikosti písma:

- **em** – Jednotka reprezentuje relativní velikost vztaženou k velikosti písma pro daný HTML element. Pokud je pomocí ní definována velikost písma, je nová velikost vztažená ke zděděné velikosti písma.
- **rem** – Jednotka reprezentuje relativní velikost vztaženou k velikosti písma kořenového elementu dokumentu. Protože je relativita konstantní pro celý dokument (nemá na ni vliv zanoření elementu), je tato jednotka ideální pro definici responzivního layoutu. Její podpora je ovšem omezena.[4]
- **ex** – Relativní velikost vztažena ke střední výšce použitého písma. Hodnota často vyjádřena vztahem 1 ex ≈ 0,5 em.
- **ch** – Jednotka udává šířku znaku 0 (nula) použitého písma.

[4] Jednotka není podporována v prohlížečích Opera Mini 8 a Internet Explorer verze 8 a starší, zdroj: http://caniuse.com/#feat=rem

Jednotky relativní k rozměrům průhledu:

- **vh** – 1/100 z výšky průhledu
- **vw** – 1/100 z šířky průhledu
- **vmin** – menší z hodnot jednotek vh a vw
- **vmax** – větší z hodnot jednotek vh a vw

Poslední a nejčastější relativní jednotkou jsou procenta (%). Hodnota procentuálního vyjádření je spočítána ze zděděné hodnoty, procenta tak mají stejnou vlastnost jako jednotka *em* - relativní vyjádření je ovlivněné mírou zanoření. [13]

3.8.1.2 Media queries

Tato specifikace přidává do CSS2 a CSS3 nové jazykové konstrukce, které dovolují definovat podmínky, za kterých se má ten který styl aplikovat. Podmínky vycházejí především z velikosti zobrazovacího zařízení a slouží k cílenému přizpůsobení vzhledu.

Media types

Specifikace vytvořená pro CSS3, nicméně i dřívější verze CSS2 a HTML4 obsahují možnost přizpůsobení pro obecný typ zařízení, např. *braille*, *handheld*, *print*, *projection*, *screen* nebo *tv*. Reálného rozšíření se dočkaly především typy *screen* a *print*. [13]

Media features

Dostupné *Media types* v CSS2 jsou velmi obecné a pro velké množství zařízení na dnešním trhnu nedostatečné, neboť nezohledňují konkrétní vlastnosti daného zařízení. V responzivním designu je důležitou vlastností šířka průhledu (*viewport*), tu ovšem žádná z předchozích definic nezohledňuje. Z tohoto důvodu byla vytvořena specifikace *Media queries*, která definuje *Media features*. To jsou nová pravidla, která jsou prohlížečem dynamicky vyhodnocována, tudíž jsou styly definovány těmito pravidly aplikovány okamžitě.

Mezi nejčastější patří:

- **(min-|max-)width** – šířka průhledu
- **(min-|max-)height** – výška průhledu
- **(min-|max-)device-width** – šířka obrazovky zařízení
- **(min-|max-)device-height** – výška obrazovky zařízení

- **orientation** – orientace zařízení (na výšku / na šířku)
- **(min-|max-)aspect-ratio** – poměr velikosti stran průhledu
- **(min-|max-)device-aspect-ratio** – poměr velikosti obrazovky zařízení
- **(min-|max-)resolution** – rozlišení zařízení; pro tuto vlastnost lze použít nový typ jednotky dpi, dpcm a dppx (viz. 3.8.1.4)
- **(min-|max-)device-pixel-ratio** – udává poměr mezi hardwarovým a CSS rozlišením

Zápis pravidel Media features je shodný se zápisem Media types, jednotlivé vlastnosti jsou uzavřeny závorkou. Klíčové slovo „and“ má význam logické spojky AND a má vyšší prioritu než logická spojka OR. [13]

Úrovně zlomu dle šířky průhledu

Úrovně zlomu určují hraniční hodnoty, od kterých se bude upravený vzhled aplikovat. Standardně se přizpůsobuje zejména rozložení a vzhled stránky na základě rozměrů cílového zařízení. Nejčastěji se vychází z dostupné šířky pro zobrazení, mnohdy také v kombinaci s vlastností *resolution*. Pokud má zařízení vyšší hustotu bodů na pixel než jedna, všechny rozměry v pixelech se dle této hustoty přepočítají. V následujícím přehledu je uvedeno obecné rozdělené zařízení dle šířky průhledu:

- **320 px** – zařízení s malou obrazovkou, telefony držené na výšku
- **480 px** – zařízení s malou obrazovkou, telefony držené na šířku
- **600 px** – menší tablety okolo 7“ (600×800 px) držený na výšku
- **800 px** – 10" tablet (768×1024 px) držený na výšku, občas se tato hranice nahrazuje **768 px**
- **1024 px** – 10“ tablet (1024×768 px) držený na šířku, notebooky, netbooky a osobní počítače
- **1200 px** – širokoúhlé displeje, především notebooky a osobní počítače

Tímto obecným dělením je možné přizpůsobit obsah webu pro zařízení s různou velikostí displeje. CSS frameworky zajišťující základní stylování webu používají 2 – 3 zlomové body, definují tak 3 – 4 cílová zařízení: mobilní telefony, tablety, osobní počítače a velké displeje. [13]

3.8.1.3 Přizpůsobení pro různou velikost průhledu displeje

Mobilní prohlížeče se od počítačů liší ve velikosti průhledu zásadním způsobem – používají průhled s rozměry velkého displeje, ale vykreslenou stránku zmenší do velikosti průhledu daného zařízení. Aby bylo možné vykreslit stránku přesně pro konkrétní průhled, uvedla společnost Apple v roce 2007 nový metatag *viewport*, kterým je možné definovat výchozí šířku průhledu, a který se v podstatě stal standardem. Může definovat následující vlastnosti:

- **width** – určuje šířku průhledu pro vykreslení; hodnotou může být velikost v pixelech (uvedená bez jednotky), nebo klíčové slovo *device-width*, které zastupuje šířku zobrazovací plochy konkrétního zařízení
- **height** – určuje výšku průhledu pro vykreslení
- **initial-scale** – výchozí přiblížení; hodnota vyjadřuje poměr šířky vykreslené webové stránky a šířky průhledu
- **maximum-scale** – maximální přiblížení; hodnota je stejného typu jako pro initial-scale
- **minimum-scale** – minimální přiblížení; hodnota je stejného typu jako pro initial-scale
- **user-scalable** – nastavení možnosti uživatelského přiblížení; hodnota je z výčtu *yes*, *no*; ve výchozím stavu je uživateli přiblížení a oddálení povoleno, pokud je hodnota *no*, jsou ignorovány vlastnosti *maximum-scale* a *minimum-scale*

Principem responzivního designu je přizpůsobení se dostupné šířce průhledu. [13]

3.8.1.4 Přepočet jednotek pro různou hustotu pixelů displeje

Nastavením výchozí šířky průhledu na dostupnou velikost je pouze relativní informací, a protože v HTML, CSS a JavaScriptu se vždy pracuje s tzv. CSS pixely, je nutné skutečné rozlišení displeje přepočítat. Velikost absolutních jednotek závisí na hustotě pixelů displeje zařízení, která lze rozdělit do dvou kategorií:

- **Zařízení s nízkým dpi** – jednotka px představuje fyzikální vlastnost výstupního zařízení, všechny ostatní jednotky jsou vůči ní definovány relativně. Mezi tato zařízení patří mobilní telefony, tablety, osobní počítače.
- **Zařízení s vysokým dpi** – jednotka px je odvozena od fyzikálních jednotek. Sem řadíme například tiskárny nebo zařízení s Retina displeji.

Hustotě pixelů odpovídá vlastnost *resolution*, pro niž má každé zařízení nastavené své vlastní hodnoty. Ty mohou být vyjádřeny v jednotkách dpi (bod na palec), dpcm (bod na centimetr) nebo dppx (bod na pixel). Pokud se *resolution* nerovná 1 dppx, jsou pixely pro prohlížeč úměrně přepočítány. Samotný přepočet je aplikován pro zachování kompatibility webů využívajících pixely pro definování velikosti.

Hustota pixelů se dá vyjádřit i poměrem mezi hardwarovým a CSS rozlišením pomocí *device-pixel-ratio*. Tato hodnota odpovídá jednotkám dppx vlastnosti *resolution*. Pomocí *Media queries* je tak možné zacílit na různá zařízení s displeji v určitém poměru i v CSS. [13] [14]

Zařízení	Úhlopříčka displeje	HW rozlišení	device-pixel-ratio	CSS rozlišení
Apple iPad 1	9,7"	1024×768	1	1024×768
Apple iPad 4	9,7"	2048×1536	2	1024×768
ASUS Nexus 7	7"	1280×800	1.325	966×604
Google Nexus 4	4,7"	1280×768	2	640×384
HTC One SV	4,3"	800×480	1.5	533×320
LG Optimus 4X HD	4,7"	1280×720	1.7	753×424
Nokia Lumia 925	4,5"	1280×768	2.4	533×320
Sony Xperia Z	5"	1920×1080	3	640×360
Xiaomi Mi3	5"	1920×1080	4	480×270

Tabulka 1: Poměr mezi HW a CSS rozlišením u vybraných zařízení, zdroj: vlastní, zdroj dat: [15] [16]

Z tabulky je možné vyčíst jednu zajímavost – i když máme 2 různá zařízení se stejně velkým displejem a shodným rozlišením, neznamená to, že na nich dosáhneme stejného zobrazení (viz rozdíl mezi Sony Xperia Z a Xiaomi Mi3).

3.8.1.5 Responzivní sazba

Font, který vypadá atraktivně na velkém displeji počítače, už tak nemusí vypadat na displeji mobilního telefonu. Proto by sazba neměla mít univerzální velikost – i typografie musí dodržovat pravidla responzivního designu.

Nejdůležitějším aspektem responzivní typografie je délka řádku. Na základě zkušeností je pro snadnou čitelnost optimální počet znaků na řádek 50 - 75 pro stolní počítače a 35 – 50 pro mobilní telefony. Sazba ovšem musí být čitelná i ve vertikálním směru, ideální velikost řádku je 140 % oproti výchozí hodnotě, u menších obrazovek by měl být mezířádkový proklad o trochu větší.

Také je vhodné zaměřit se na druh písma, některé módní fonty vypadají vzhledově příjemně na velké obrazovce, ale na menší obrazovce může nastat problém s jejich čitelností, protože tyto druhy písem potřebují dostatečné mezery. Pracujeme-li s malým prostorem, je vhodnější použít bezpatkové písmo. [12]

3.8.2 Responzivní obrázky

V responzivním designu často potřebujeme volit mezi různými variantami jednoho obsahového obrázku. Nejčastěji proto, že chceme ušetřit datový objem stránky na mobilních telefonech. Iniciativa RICG[5] tak vymyslela 9 scénářů, kdy je potřeba jeden obrázek reprezentovat více různými variantami, nejdůležitější jsou ovšem tyto čtyři:

- Výběr varianty obrázku podle velikosti okna prohlížeče.
- Výběr varianty podle velikosti obrázku v rámci layoutu stránky. V některých situacích totiž můžeme potřebovat na větším displeji menší obrázky (například na tabletu zobrazím více menších obrázků v mřížce, ale na mobilním telefonu větší obrázky v jednom sloupci).
- Výběr podle poměru mezi hardwarovým a CSS rozlišením (viz. 3.8.1.4).
- Výběr podle art direction (výtvarné režie), tj. pokud chceme obrázek na různých zařízeních jinak oříznout.

Pro tyto účely byly do tagu <img> zavedeny nové atributy, které umožní zobrazovat v různých stavech responzivního designu různé varianty obrázků. Jedná se o atributy *srcset* a *sizes* a výhodou je na nich fakt, že poměrně složité rozhodování jaký obrázek v dané situaci použít obstarává prohlížeč. Autor webových stránek pouze určí, jaké varianty obrázku má k dispozici (*srcset*) a jak jsou velké mezi jednotlivými zlomy layoutu (*sizes*).

```
<img src="small.png"
srcset="small.png 600w, medium.png 1024w, large.png 1600w"
sizes="(min-width: 768px) 300px, 100vw"
alt="Obrázek" width="200" height="200">
```

Ukázka kódu 1: Definice responzivního obrázku 1, zdroj: [17]

V uvedeném příkladu prohlížeči sdělujeme, že máme k dispozici obrázek small.png o šířce 600 px, medium.png o šířce 1024 px a large.png o šířce 1600 px.

[5] Responsive Images Community Group

Pokud by nebyl uvedený atribut *sizes*, prohlížeč by se při výběru obrázku k zobrazení rozhodoval podle šířky okna. Do 600 px a méně širokého okna by načetl small.png, mezi 601 a 1024 px pak medium.png a pro okno široké 1025 px a více by načetl large.png. Prohlížeč bere v potaz i aktuální device-pixel-ratio (tj. poměr mezi HW a CSS rozlišením), takže pro ratio = 2 načte medium.png i v případě, že okno je široké 600 px.

V praxi je ale mnohdy potřeba zobrazovat obrázek nikoliv podle šířky okna, ale podle šířky obrázku v rámci layoutu. K tomu slouží atribut *sizes*. V uvedeném příkladu zápis udává, že v rozlišeních 768 px a více má mít obrázek 300 px, jinak 100 % šířky průhledu.

V atributu *sizes* je ovšem možné použít také funkci *calc()*, díky které můžeme přesně definovat velikost obrázku relativně k layoutu mezi jednotlivými zlomy. To se hodí v případě, že přesně nevíme, jaké rozměry budou mít obrázky v rámci konkrétní šířky okna.

```
<img src="small_600.png"
srcset="small_600.png 600w, medium_1024.png 1024w, large_1600.png 1600w"
sizes="(min-width: 600px) calc((100vw - 2*8px) * 0.49), calc(100vw - 2*8px)"
alt="Obrázek" width="200" height="200">
```

Ukázka kódu 2: Definice responzivního obrázku 2, zdroj: [17]

V tomto příkladu budou velikosti přeloženy takto:

- V rozlišení 600 px a více bude mít obrázek velikost (100 % šířky průhledu – 8 px po stranách) * 49 % šířky obrázku.
- V rozlišení 599 px a méně bude mít obrázek velikost 100 % šířky průhledu – 8 px po stranách.

Atributy *srcset* a *sizes* vystačí v naprosté většině situací, kdy je potřebné řešit responzivní obrázky. Nevýhodou tohoto řešení je stávající podpora v prohlížečích, nativní podpora je v posledních verzích Chrome, Opery a částečně Safari. Do budoucna je však slíbena podpora ve všech prohlížečích. [17]

4 Analýza firmy

Tato část vývoje informačního systému je velmi důležitá, protože chyby, které se neodhalí během analytické fáze, se ve většině případů přenesou až do fáze vývoje a v tu chvíli je již velmi obtížné je řešit a odstraňovat. Proto by měl být kladen velký důraz na důkladnou analýzu všech požadavků a potřeb klienta.

Chyby, které se přenesou až do finálního produktu, jsou odstranitelné jen za cenu velkých časových nebo finančních ztrát, což nechce ani jedna ze zúčastněných stran podstupovat.

4.1 Základní údaje

V roli klienta při vývoji informačního systému v této práci vystupuje mladá firma, která se zabývá konstrukcí 3D tiskáren a zakázkovým tiskem výrobků na 3D tiskárně. Firma působí na trhu velmi krátce, stále se zabývá především kusovou a malosériovou výrobou. Proto je pro ni velmi důležité soustředit se na její předmět podnikání a ostatní procesy si co nejvíce zjednodušit. Firma by se tak měla soustředit především na kvalitu vlastní produkce, budování jména a především dobrých vztahů se svými klienty. K tomu by jí měl dopomoci na míru stavěný informační systém.

Se systémem budou v začátku pracovat pouze dva uživatelé, proto bude v některých ohledech zjednodušen a do budoucna se bude počítat s dalšími úpravami a rozšířením. Nízkému počtu uživatelů a požadavku na nízkou cenu bude také odpovídat návrh komponent pro stavbu serveru.

4.1.1 Vize a cíle firmy

V současné době se firma soustředí především na zakázkový tisk různých modelů a jiných výrobků. Potenciál ovšem cítí také v tisku univerzálních součástek, například právě pro stavbu dalších 3D tiskáren. Pro tyto účely by se ve firmě hodil jednoduchý nástroj pro snadnou správu jednotlivých výrobků i s možností nahrávání fotografií. Bylo by také vhodné, pokud by z tohoto nástroje vznikl jednotný katalog, kterým by se firma mohla prezentovat stávajícím, ale především také budoucím klientům. Firma do budoucna také zvažuje nasazení eshopu, ve kterém by mohla nabízet ukázkové a univerzální výrobky, součástky pro konstrukci 3D tiskáren, případně i hotové tiskárny.

4.1.2 Stávající portfolio firmy

Jak již bylo zmíněno, firma dokáže vyrobit téměř jakýkoliv myslitelný tvar nebo model. Technologie 3D tisku má sice stále určitá omezení (například ve velikosti v kuse tisknutelného výrobku), ale stále se vyvíjí a do budoucna jí je přikládán velký význam.

Konkrétními příklady portfolia mohou být druhá tiskárna vytištěná na první tiskárně, driftovací „pneumatiky“ pro R/C model, klíčenka pro firmu KOOPEX, model BMX kola, koncovky na řídítka, náhrada ozubeného kolečka, vlastní vizitky, model motoru VR6, model pyramidy, postavička Yody, soška Sfingy a další.

4.2 SWOT analýza

Silné stránky

- Nadšený mladý tým bez závazků
- Vlastní konstrukce 3D tiskáren
- Zkušení pracovníci – konstrukce, modelování
- Prostorné kancelářské prostory
- Téměř neomezená pracovní doba
- Schopnost zpracovat jakékoliv zadání a vyrobit pro něj 3D tiskárnu

Slabé stránky

- Nedokonalé firemní procesy
- Chybějící informační systém
- S tím související chybějící jednotná podniková směrnice
- Celková nezkušenost v podnikání
- Vlastní konstrukce 3D tiskáren

Příležitosti

- Spuštění vlastního eshopu
- Výroba materiálu pro 3D tisk
- Výroba 3D tiskáren
- Čistě konstrukční kancelář, bez výrobního aparátu

Hrozby

- Odchod investora
- Opuštění poptávky po 3D tisku
- Vykradení, vyhoření nebo jiné zničení kancelářských prostor
- Ztráta nadšení

4.3 PEST analýza

Jak bylo zmíněno v kapitole 3.4.2, PEST analýza hodnotí pouze vnější faktory podniku, které nemůže svými aktivitami, nebo jen velmi obtížně, zvládnout.

4.3.1 Politické prostředí

Stejně jako každá firma podnikající na území České republiky se musí i tato řídit platnými zákony, předpisy nebo nařízeními. Mezi ty, jež podnikání ovlivňují pozitivně nebo negativně, patří především:

- Zákon č. 89/2012 Sb., občanský zákoník – upravuje právnické osoby, formy podnikání, obchodní závazkové vztahy, právní skutečnosti atd.
- Zákon č. 90/2012 Sb., o obchodních korporacích – definuje obchodní korporace, obecná ustanovení atd.
- Zákon č. 586/1992 Sb. o daních z příjmů a ve znění pozdějších předpisů – stanoví podnikatelskému subjektu, jak velkou část zisku má odvést do státního rozpočtu.
- Zákon č. 235/2004 Sb. o dani z přidané hodnoty a ve znění pozdějších předpisů.
- Zákon č. 563/1991 Sb. o účetnictví a ve znění pozdějších předpisů – tento zákon stanovuje firmě rozsah a způsob vedení účetnictví s požadavky na jeho průkaznost.
- České účetní standardy pro účetní jednotky, které účtují podle vyhlášky č. 500/2002 Sb. a ve znění pozdějších předpisů – účetní standarty mají docílit souladu při používání účetních metod účetními
- Zákon č. 262/2006 Sb. zákoník práce a ve znění pozdějších předpisů – vymezuje vztahy na pracovišti mezi zaměstnavatelem a zaměstnancem.

Vlastní podnikání ovlivňují další faktory, především daň z přidané hodnoty. Její současná výše je 21 % pro základní sazbu a 15 % pro sníženou sazbu vešla v platnost v roce 2013. V roce 2015 vešla v platnost druhá snížená sazba – 10 %. Se zvyšováním DPH bohužel

souvisí i zdražování vstupů a díky tomu i výstupů firmy. Naopak vývoj daně z příjmu právnických osob má pozitivní vliv pro podnikání firmy, neboť se od roku 2000 snížila až na současných 19%.

4.3.2 Ekonomické prostředí

Mezi základní ukazatele, které ovlivňují podnikání firmy, patří hrubý domácí produkt, míra inflace, vývoj úrokových sazeb, mezd, nebo nezaměstnanosti.

Kupní síla pro tuzemské produkty je vzhledem k rostoucímu kurzu koruny vůči euru a dolaru poměrně vysoká, úrokové sazby a inflace jsou nízké, což nahrává především možnému rozšiřování firmy a nabízeného portfolia. Příznivý vliv na firmu má i rostoucí hrubý národní produkt. Navíc cena energií se za poledních několik let ustálila na jedné hladině, ale v porovnání s cenou strojů je toto zanedbatelná položka. Na druhou stranu drahé zahraniční měny ztěžují nákup materiálu na čínském trhu. Z dlouhodobého hlediska je také nevýhodné vysoké daňové zatížení zaměstnanců.

4.3.3 Sociální prostředí

Demografický vývoj populace značí zvětšování zákaznické základny vzhledem k nárůstu počtu narozených dětí. Mobilita je na území ČR dobrá, díky tomu se mohou zákazníci dopravit přímo do firmy, stejně tak není problém pro zástupce firmy se dopravit přímo k zákazníkovi.

Dále se zástupci firmy domnívají, že v souvislosti s nárůstem počtu obyvatel se lidé začínají více zajímat o to, jaký mají životní cyklus.

4.3.4 Technologické prostředí

Jedním z ukazatelů pro hodnocení technologického prostředí je možné využít výdaje státního rozpočtu na vývoj a výzkum. Výhodou je i členství České republiky v Evropské unii. Předpokládá se, že výše investic EU do tohoto odvětví bude přibližně 120 miliard Kč, výše investic na jeden projekt pak od 500 milionů Kč.

Dalším ukazatelem je stav ICT a dalších technologií, které firma využívá. Aktuální technologii hodnotí zástupci firmy jako lehce zastaralou, nicméně vzhledem k investicím EU toto nevidí jako velkou překážku. V oblasti 3D tisku navíc dochází stále k novým objevům a díky otevřenosti kolektivu i k větším změnám je výhled do budoucna optimistický.

4.4 Aktuální stav a požadavky firmy

Tato kapitola se zaměřuje na popis současného SW a HW vybavení ve firmě, které slouží k provozu firmy. Součástí jsou také požadavky zástupců firmy na funkčnost plánovaného informačního systému.

4.4.1 Současné vybavení

Z pohledu software má firma obrovský potenciál, neboť v tuto chvíli nepoužívá žádný informační systém. Vývoj tak není omezen na zachování zpětné kompatibility s existujícím systémem, případně na nutnost tvorby rozhraní pro import dat z nějakého specifického formátu.

Na pracovních stanicích jsou nainstalovány operační systémy od společnosti Microsoft, konkrétně Windows ve verzi Vista a 7. Mezi primární prohlížeče internetu patří Google Chrome, Opera a Vivaldi. Všechny tyto prohlížeče používají v nejnovějších verzích jádro Blink. Jedná se o Open Source jádro vyvinuté společností Google v rámci projektu Chromium.

Na dalších zařízeních (mobilní telefony a tablety), která jsou ve firmě využívána, a na kterých se plánuje využívání informačního systému, běží operační systém Android. Pro prohlížení webu pak slouží jak vestavěný prohlížeč, tak i Opera Mini.

Z dalšího softwarového vybavení stojí za zmínku kancelářský balík Open Office nebo různé modelovací nástroje pro tvorbu podkladů pro 3D tisk, jmenovitě například FreeCAD.

Mezi hlavní hardwarové platformy, které se ve firmě nacházejí, lze řadit především dva desktopové počítače. K nim jsou připojeny tiskárny, a také na nich probíhá tvorba nových výrobků v modelovacím softwaru. Dále sem pak patří jeden notebook, dva mobilní telefony a dva tablety.

Používaná rozlišení, pro která se bude systém primárně tvořit a optimalizovat:

Zařízení	Rozměr obrazovky	Rozlišení
Stolní počítač	19“	1440x900
Stolní počítač	19“	1280x1024
Notebook	14“	1280x800
Tablet Samsung Galaxy Note 9	8,9“	1280x800
Tablet Google Nexus 7	7“	1280x800
MT Lenovo P780	5“	1280x720
MT HTC One SV	4,3“	800x480

Tabulka 2: Seznam používaných zařízení ve firmě, zdroj dat: analýza firmy

4.4.2 Požadavky firmy

Protože firma neměla konkrétní představu o informačním systému, bylo před započetím celkové analýzy a vlastních prací na systému uskutečněno několik informačních schůzek, na kterých se postupně řešily jednotlivé požadavky na informační systém, ale také náročnost realizace, priorita jednotlivých požadavků a také budoucí provoz.

Celkové shrnutí požadavků podle priority od nejdůležitějších:

- Online provoz, dostupnost odkudkoliv, kde je k dispozici internetové připojení
- Responzivní vzhled – systém bude využíván na stolních počítačích, ale i na smartphonech
- Kompletní správa a evidence objednávek a faktur – primární účel
- Správa zákazníků
    - Mít možnost evidence více doručovacích adres u jednotlivých zákazníků
    - Zároveň slouží jako adresář
- Skladová evidence materiálu, který se používá pro tisk modelů
    - Evidence materiálů, barev
    - Skladová dostupnost materiálů
    - Možnost změny ceny materiálu pro konkrétní složitost výrobku
- Evidence dokončených výrobků v rámci otevřené objednávky
- Správa 3D modelů, jejich navázání přímo na objednávky
    - Z této evidence by mělo být možné vytvořit online katalog
    - Fyzická přítomnost modelů na serveru – systém bude sloužit jako jednotné úložiště

- Kalendář pro evidenci úkolů, schůzek nebo poznámek
- Hlavní obrazovka se zobrazením nejdůležitějších informací, mezi které patří naplánované schůzky a aktuálně zpracovávané objednávky
- Přístup uživatelů do systému na bázi uživatelských rolí
- Napojení na emailovou službu
    - Informování klientů o hotových zakázkách
    - Informování uživatelů o nadcházejících událostech, úkolech a podobně
- Jednoduchý a přehledný vzhled

4.5 Hlavní podnikové procesy

Pro detailní popis a modelování pomocí EPC diagramů byly zvoleny dva nejdůležitější firemní procesy – práce s objednávkou a fakturace objednávky.

Z modelů lze snadno vyčíst, který pracovník firmy je za jednotlivé dílčí procesy zodpovědný, a které části informačního systému jsou v těchto procesech využívány.

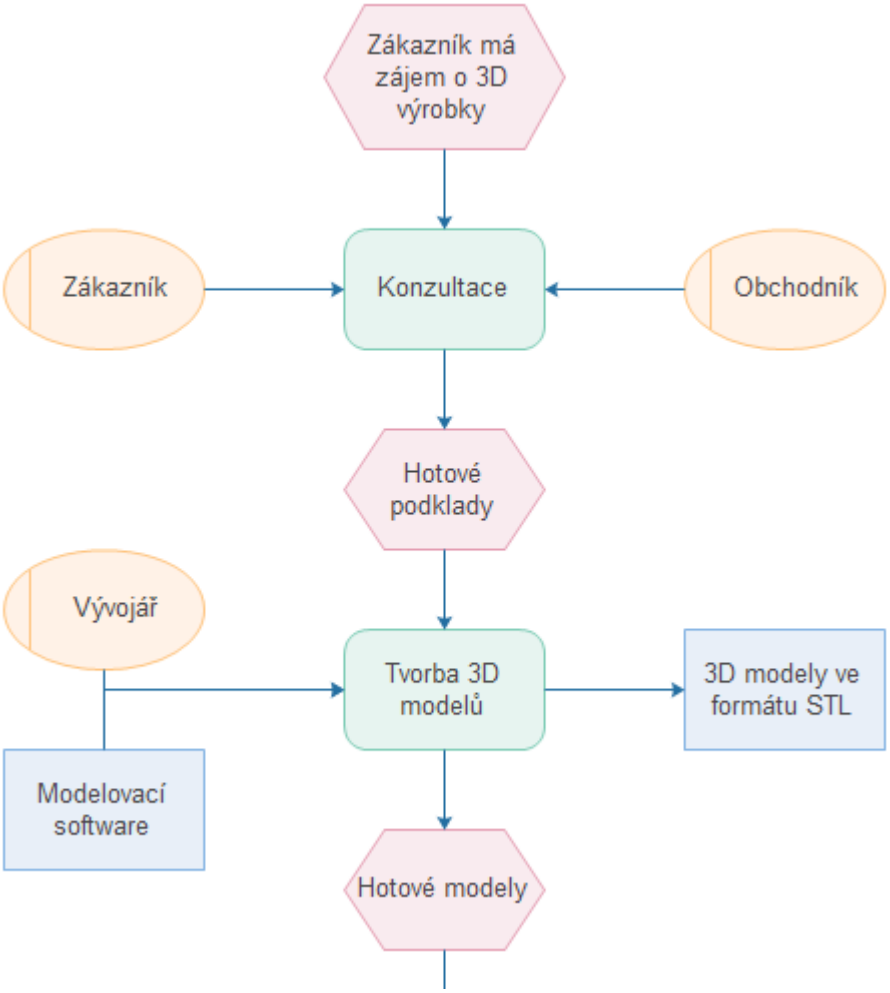
K tvorbě EPC diagramů byl využit program Edraw Max.

4.5.1 Životní cyklus objednávky

Založení vlastní objednávky předcházejí konzultace požadavků se zákazníkem. Na základě těchto požadavků vzniká 3D model, který vystupuje v rámci objednávky jako její položka. Z 3D modelu je zjištěna materiálová náročnost, ze které se odvíjí cena za každý kus a tím i cena celé objednávky.

Pro přehlednost je každý vytištěný model zanesen do objednávky. Ve chvíli, kdy jsou vytištěné všechny modely, je objednávka označena za zpracovanou a zákazník je o tomto kroku informován emailem. Současně je vygenerována faktura, kterou je nutné zkontrolovat a schválit, čímž se objednávka označí za vyfakturovanou a je možné ji předat/odeslat zákazníkovi.

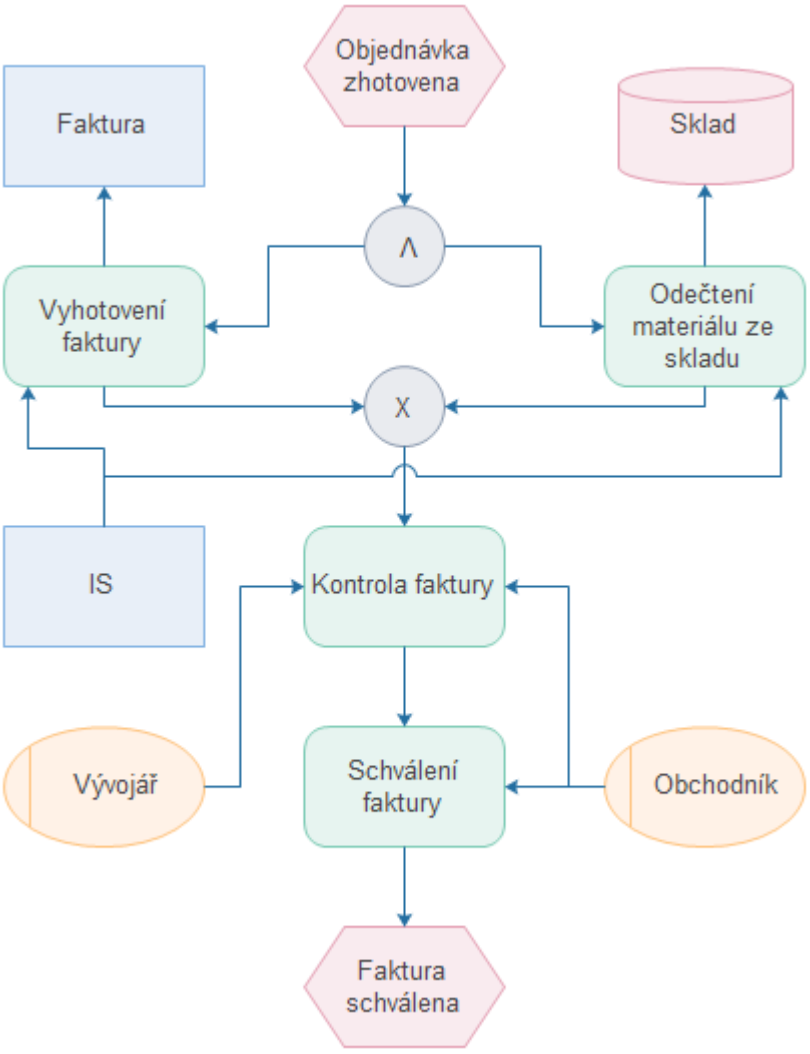
Kvůli velikosti celého diagramu je zde pouze jeho část, kompletní diagram se nachází v příloze.



Obrázek 10: Zkrácený EPC diagram životního cyklu objednávky, zdroj: vlastní

4.5.2 Fakturace objednávky

Po dokončení objednávky je založena faktura a blokovaný materiál je odečten ze skladové dostupnosti. Jakmile je faktura schválena, na základě vybraného způsobu dodání pracovník buď kontaktuje dopravce, nebo přidá do kalendáře předání zhotovené objednávky zákazníkovi.



Obrázek 11: EPC diagram fakturace objednávky, zdroj: vlastní

4.6 Zhodnocení analýzy

Z hlavního požadavku je jasné, že firma chce online řešení, které bude fungovat na různorodých zařízeních. To znamená optimalizace z hlediska rozlišení, ale také podpory jednotlivých technologií v různých verzích internetových prohlížečů. Další požadavky a informace již poskytly dostatek informací, které byly využity při vlastním návrhu, a kterými se implementace informačního systému řídila.

První fáze návrhu informačního systému ovšem byla poněkud ztížena faktem, že firma neměla přesně nastavené jednotlivé firemní procesy a proto nebylo možné se vždy spolehlivě opřít o zásady práce, jakými se pracovníci řídí. Nicméně v průběhu konzultací byly postupně dohodnuty postupy pro nejdůležitější firemní procesy a nic tak nebránilo dokončit celkový návrh a začít s vlastní implementací.

Celý systém se tak tvořil na zakázku a přesně pro potřeby firmy na základě dohodnutých požadavků.

5 Vlastní řešení

Tato kapitola se zabývá vlastním řešením informačního systému. Popisuje použité technologie a ve zkratce zdůvodní, proč byly vybrány. Dále obsahuje popis jednotlivých modulů informačního systému a také návrhy uživatelského rozhraní, drátěných modelů a finální vizualizaci na různých testovacích zařízeních.

5.1 Použité technologie

Informační systému bude poskytován jako webová služba, této skutečnosti odpovídají použité technologie při jeho implementaci. Nejdůležitější součástí je Yii Framework, který posloužil k tvorbě serverové části aplikace.

5.1.1 HTML, PHP, CSS

Technologie jako PHP, HTML nebo CSS jistě není třeba zdlouhavě popisovat, proto budou popsány pouze ve zkratce.

5.1.1.1 HTML

HyperText Markup Language je značkovací jazyk, který se využívá k vytváření základní obsahové struktury webových stránek. Jazyk HTML je charakterizován množinou značek (tagů) a jejich vlastností (atributů), které jsou definovány v jazyce SGML (metajazyk, který slouží pro definici jiných jazyků). První verze byla navržena v roce 1990 ve výzkumném centru CERN ve Švýcarsku společně s HTTP protokolem.

Mezi jednotlivé HTML tagy se vkládá text a tím se definuje obsah stránky. Dříve sloužil HTML jazyk také k formátování vzhledu stránek. Od tohoto způsobu se v minulosti kvůli zachování přístupnosti webu upustilo a začaly se k tomuto účelu využívat CSS. Ty umožňují vytvářet vzhled jako samostatnou vrstvu webové stránky. [18]

```
<html>
<head>
	<title>Jednoduchá stránka</title>
</head>
<body>
	<h1>Nadpis</h1>
	<p>Text text text text text text text text text text text</p>
</body>
</html>
```

Ukázka kódu 3: Ukázka jednoduché webové stránky, zdroj: vlastní

5.1.1.2 PHP

Dynamické stránky jsou v současné době nezbytnou součástí každé složitější internetové prezentace. Mezi hlavní skriptovací jazyky, které se k tvorbě těchto stránek využívají, patří právě PHP.

PHP (rekurzivní zkratka pro PHP: Hypertext Preprocesor) je skriptovací jazyk, který se vkládá do běžného HTML kódu. Na rozdíl od jiných skriptovacích jazyků (jako například JavaScript) je činnost zdrojového kódu prováděna na straně serveru. To znamená, že server vezme celou stránku, vykoná všechny části PHP kódu a na straně klienta se vykreslí čisté HTML, které je výsledkem provedeného PHP skriptu. Uživatel se tak nedozví, jakým způsobem byla stránka vytvořena.

První verze PHP byla vydána 8. června 1995 dánským programátorem Rasmusem Lerdorfem. V roce 1998 bylo přepsáno jádro PHP a vzniklo tak nové jádro – Zend. V roce 2004 bylo vydáno PHP5 na jádře Zend Engine 2. Tato verze přinesla zásadní změnu v jazyce PHP – možnost objektově orientovaného programování. Aktuální stabilní verze je 5.6.4 vydaná 18. prosince 2014.

5.1.1.3 CSS

Kaskádové styly (z anglického Cascading Style Sheets) jsou moderním jazykem, který umožňuje jednoduché a účinné formátování webových stránek. Hlavní vlastnost, kterou mají styly v názvu, je kaskádovost – tedy jednotlivé styly se mohou překrývat, což zvyšuje jejich efektivnost. Pokud jsou CSS správně použity, dojde k oddělení vzhledu stránky od jejího obsahu. Oddělením prezentační vrstvy je zvyšována přístupnost webu.

Výhody využití CSS:

- Větší možnosti formátování
- Rychlejší načítání stánek a menší zátěž serveru
- Použitím CSS šablon lze snáz upravovat větší stránky

Nevýhody:

- Ne vždy 100% podpora v prohlížečích (především starší verze Internet Exploreru)
- CSS selektory neumožňují přístup k rodičovským elementům

- CSS neumějí počítat výrazy (nelze tedy sčítat rozměry a podobně)

Dne 7. června 2011 byla vydána dnes asi nejrozšířenější verze – CSS 2.1, ovšem s rozšiřováním standardu HTML5 a moderních webových prohlížečů je stále více využíváno aktuální verze CSS 3.

Opravdová síla CSS se nejlépe projeví při využití některého z CSS frameworků (například Bootstrap, kapitola 5.1.3) nebo preprocesorů jako například SASS nebo LESS (kapitola 5.1.5). Preprocesory navíc smazávají některé nevýhody, jako například absenci počítání výrazů. [19]

5.1.2 Yii2 framework

Yii je výkonný PHP Framework[6] určený pro snadnou tvorbu webových aplikací, aktuální verze nese číslovku 2.0.3[7], proto označení Yii2. Název Yii je zkratkou pro „*Yes It Is!*“. Autoři frameworku tuto zkratku uvádějí jako tu nejpřesnější a nejvýstižnější odpověď na otázky nováčků, typicky *„Is it fast? ... Is it secure? ... Is it professional? ... Is it right for my next project?“ … „Yes, it is!“*.

Yii je svobodný, open-source framework napsaný v PHP5, který podporuje čistý návrh aplikací s DRY (viz kapitola 5.1.2.2) přístupem a především jejich rychlý vývoj. Snahou Yii je co nejvíce zefektivnit vývoj a vytvořit maximálně efektivní, rozšiřitelné a udržovatelné aplikace. Obsahuje navíc různé nástroje na podporu testování a ladění aplikace a má také rozsáhlou a ucelenou dokumentaci.

Vývoj frameworku začal 1. ledna 2008 vývojářem Qiang Xue, který dříve vyvíjel a spravoval framework Prado, a první stabilní verze, Yii 1.0, byla vydána 3. prosince 2008. Prado je pro Yii hlavním zdrojem nápadů. Čerpá z něho například vrstvu databázové abstrakce, modulární architekturu, internacionalizaci a mnoho dalších funkcí a návrhů. Mezi další zdroje inspirace patří Ruby on Rails, Symfony, Joomla a jQuery, které Yii integruje jako základní JavaScriptový framework.

Tato kapitola a její podkapitoly byly zpracovány s pomocí zdroje [20], při vlastním programování bylo hojně využíváno online dokumentace [21] a zdroje [22].

[6] Framework je softwarová struktura, která slouží jako podpora při programování. Obsahuje podpůrné programy, knihovny, API nebo doporučené postupy při vývoji.
[7] Vydaná 1. března 2015

5.1.2.1 Funkce frameworku

Tato kapitola se zabývá jednotlivými funkcemi frameworku, kterými zjednodušuje vývojáři práci a ten se díky tomu může soustředit na specifické úkoly.

MVC design

MVC design, neboli zkratka pro Model-View-Controller, je široce používaný přístup při vývoji webových aplikací. Pokud se omezíme na zjednodušující popis, tak MVC odděluje business logiku aplikace od uživatelského prostředí, takže vývojáři mohou jednodušeji upravovat jednotlivé části bez vlivu na ostatní. V MVC architektuře model představuje data, view obsahuje prvky uživatelského rozhraní a controller zajišťuje komunikaci mezi view a modelem.

Vedle implementace MVC architektury využívá Yii komponentu Application, která vystupuje jako předek controlleru a zajišťuje zpracování HTTP requestu. Tato komponenta shromáždí veškeré informace o requestu uživatele a odešle je do vhodného controlleru pro další zpracování.

Vrstva databázové abstrakce

Yii obsahuje vrstvu pro práci s databázemi postavenou na PHP rozšíření s názvem PHP Data Objects (PDO) – Yii Data Access Objects (DAO). Tato vrstva umožňuje přístup k různým databázovým systémům skrze jednotné rozhraní. Aplikace využívající tuto vrstvu mohou být snadno migrovány na různé databázové systémy, aniž by bylo nutné nějak upravovat zdrojový kód.

Další komponenta, Yii Query Builder, nabízí objektový přístup pro sestavování databázových dotazů a pomáhá tak snižovat riziko SQL injection.

Yii Active Record je implementací objektově relačního mapování a ještě více zjednodušuje programování na databázové vrstvě. AR představuje tabulku v podobě třídy a řádky tabulky jako jednotlivé instance této třídy. Yii tak pomáhá odstraňovat opakované činnosti při psaní SQL dotazů, které se týkají především CRUD[8] operací.

Formulářové prvky a validace

Kromě tvorby formulářů a jejich polí musí vývojář řešit i další úkony, jakými jsou například naplnění formulářových polí existujícími daty nebo defaultními hodnotami,

[8] Create, read, update, delete, základní operace s entitami

validace uživatelského vstupu, zobrazení chybových hlášek nebo ukládání do některého z persistentních datových úložišť.

Tomu Yii dopomáhá svou MVC architekturou:

- Model se svými atributy představuje formulářové prvky a data, která se budou od uživatele získávat.
- Controller sbírá data z formuláře a předává je modelu.
- View v odpovídající akci controlleru vykresluje vlastní formulář.

Validaci vlastních dat zajišťuje metoda modelu, její výsledek závisí na tom, zda se všechny vstupy podařilo zvalidovat, či nikoliv. Yii v základu disponuje řadou vestavěných validátorů pro kontrolu vstupu, například zda se jedná o číslo, datum, email, URL a další. Pokud by vývojáři vestavěná pravidla nestačila, může si naprogramovat vlastní, ať již v podobě metod v rámci jednoho modelu, nebo v podobě vlastní validační třídy, kterou je možné použít univerzálně pro více modelů. Žádná moderní webová aplikace se neobejde bez validace vstupu na straně klienta, i tuto funkcionalitu Yii framework podporuje, ať již pomocí jednoduchých vestavěných validátorů nebo pomocí AJAXové validace.

AJAXové widgety

Protože Yii ve své instalaci integruje jQuery, už v základu disponuje sadou AJAXových widgetů. Patří mezi ně dokončovací formulářová pole, stromová zobrazení, datagridy a další. To vše umožňuje vytvářet efektivní a všestranná uživatelská rozhraní.

Autentifikace a autorizace

Proces autentifikace a autorizace je nutný pro všechny (nejen) webové stránky a aplikace obsahující data, která se mají zobrazovat jen určitým uživatelům a za určitých podmínek. Typickým příkladem je právě systém, kterým se zabývá tato práce. Autentifikace slouží k ověření, zda je uživatel skutečně tím, za koho se vydává. Ta obvykle zahrnuje uživatelské jméno a heslo, ale může být provedena i jinou metodou, třeba otiskem prstu. Autorizace slouží k ověření, zda přihlášená osoba má oprávnění číst stránku nebo manipulovat s daty.

Způsob implementace vlastního procesu autentifikace je na vývojáři, Yii mu k tomu poskytuje rozhraní IdentityInterface s metodami, které jsou pro tento proces vyžadovány.

Pro autorizaci uživatele nabízí Yii dva nástroje. Jednoduchý Access Control Filter využijí aplikace, kde není potřeba složitá kontrola přístupu, protože dokáže pracovat pouze s dvěma rolemi – user (@) a guest (?). Složitějším, ale mnohem mocnějším nástrojem, je Role-Based Access Control. Pro jeho funkčnost je nutné splnit dvě podmínky – v aplikaci musí být přítomna autorizační data (tj. seznam rolí, akcí a jejich provázání na uživatele) a aplikace musí tato data využívat v místech, kde jsou zapotřebí.

Skiny a šablony

Yii implementuje mechanismus pro snadné šablonování vykreslovaných stránek. V aplikaci je možné mít nadefinováno nespočet různých vzhledů, a protože se aplikace samotného layoutu provádí až na konci běhu controlleru, je možné měnit vzhled dynamicky, například na základě nastavení uživatele.

Yii také podporuje dva šablonovací enginy, Twig a Smarty. Výhodou těchto šablonovacích enginů je jednotná syntaxe a zamezení vývojáři v provádění výkonného kódu v rámci view, kam vůbec nepatří.

Webové služby

Pro tvorbu webových služeb nabízí Yii sadu nástrojů a vlastností, které zjednodušují implementace tzv. RESTful API. Pro každý typ zdroje (např. uživatel, objednávka) se definuje vlastní controller a koncové body webové služby jsou implementovány pomocí jednotlivých akcí controlleru. Zdroje jsou reprezentovány datovými modely, a pokud vývojář pracuje s databázemi, je vhodné použít pro tyto účely ActiveRecordů, ale není to podmínkou.

Internacionalizace a lokalizace

Pro podporu překladů webové aplikace definuje Yii dva typy jazyků – zdrojový a cílový. Zdrojový jazyk je ten, ve kterém jsou texty napsány přímo do zdrojového kódu pomocí překladových funkcí. Cílovým jazykem se myslí takový, ve kterém si uživatel zobrazuje webovou aplikaci. Do něho musí systém texty přeložit, k čemuž mu slouží překladové slovníky.

Cachování

Cachování může probíhat na různých úrovních a místech webové aplikace. Na straně serveru, tedy nižší vrstvě, může být cache použita pro ukládání základních dat. Na vyšší úrovni pak cache může sloužit pro ukládání celých fragmentů webových stránek. Na klientské straně může být HTTP cache využita ke skladování obsahu nedávno navštívených stránek.

Yii podporuje všechny druhy cachovacích mechanismů – data cache, fragment cache, page cache a HTTP cache.

Error handling a logování

Yii obsahuje vestavěný error handler, který se stará o správné a mnohem příjemnější odchytávání chyb. Toho dociluje pomocí následující funkcionality:

- Všechny non-fatal PHP errory (tj. warning a notice) převádí na výjimky, které je možné následně odchytit.
- V ladícím módu jsou výjimky a fatal PHP errory zobrazeny s detailními informacemi a částmi zdrojového kódu.
- Využívá vlastní controller a action pro zobrazování chyb.
- Podporuje různé formáty HTTP odpovědi pro zobrazení chyb.

Zpracování chyb je ovšem pouze jedna strana mince, pokud si totiž aplikace chybu zpracuje, ale nikam ji na pozadí nezapíše, nemá se vývojář jak dozvědět, že k ní vůbec došlo. Yii k tomuto účelu nabízí několik nástrojů, kterými je možné zapisovat chyby různých stupňů do souboru, databáze, nebo je odesílat emailem na konkrétní příjemce.

Bezpečnost

Každý moderní framework by měl poskytovat nástroje pro zvýšení bezpečnosti aplikací, které v něm jsou vyvíjeny. Yii není v tomto ohledu výjimkou.

V první řadě by měl uživatel myslet na ošetření SQL injection. To je velice zjednodušeno ve chvíli, kdy používá DAO, Query Builder nebo Active Record pro práci s databází. Pokud vývojář skládá SQL dotazy ručně, měl by pro vkládání hodnot do dotazu využít k tomu určené funkce DB vrstvy, které vstup automaticky ošetřují.

Dalším bezpečnostním rizikem je XSS, neboli Cross-site scripting. Jeho ošetření mají zjednodušené vývojáři, kteří využívají některého z šablonovacích enginů, protože ty ošetřují výstupy z funkcí a proměnných již při vykreslování. Pokud vývojář používá standardní PHP šablony, měl by výstupní data ošetřit ručně pomocí vestavěných tříd Html nebo HtmlPurifier.

Další bezpečnostní rizika, jako například CSRF nebo povolené procházení souborů na serveru, nejsou specifická pro daný framework, o to spíše by na ně měl vývojář myslet.

Automatické generování kódu

Gii je velice užitečný nástroj, který vývojáři umožňuje generovat nejčastěji používaný zdrojový kód, stejně tak jako kompletní CRUD controllery. Velkou výhodou Gii je jeho ovládání přes webové rozhraní pro interaktivní generování kódu, který vývojář vyžaduje. Pro vývojáře, kteří naopak upřednostňují ovládání systému přes konzoli, obsahuje Gii také rozhraní pro příkazový řádek.

Mezi hlavní generátory, které je možné v Gii používat, patří tyto:

- Model generator – vytváří ActiveRecord pro tabulku v databázi
- CRUD generator – vytváří controller a view pro CRUD operace nad zadaným datovým modelem
- Controller generator – ze zadaného názvu a metod vytvoří kompletní controller s akcemi a view
- Form generator – vygeneruje view soubor s formulářem ze zadané třídy

Podpora kódu třetích stran

Každý vývojář potřebuje čas od času použít knihovny třetích stran, ať již z důvodu chybějící funkcionality ve frameworku, nebo prostě jen z toho důvodu, že je zvyklý danou knihovnu používat. Na tyto situace Yii také myslí a vývojáři stačí zajistit správný include knihovny do aplikace nebo její načtení pomocí autoloadu.

Detailní dokumentace

Každá metoda nebo vlastnost frameworku je detailně a jasně popsána v rozsáhlé dokumentaci. K dispozici jsou také tutoriály nebo tištěné knihy, které pomáhají lépe pochopit jednotlivé funkce frameworku.

Knihovna rozšíření

Díky rozsáhlé komunitě okolo Yii je na oficiálních stránkách velké množství knihoven, které pomáhají rozšířit a vylepšit funkcionalitu frameworku.

Hlavním důvodem, proč bylo využito Yii frameworku, je přibližně dvouletá aktivní zkušenost s programováním v tomto frameworku, jak ve verzi 1.x, tak i v poslední verzi 2.0. Zvažován byl i Nette framework, který, ač s ohromnou popularitou[9], nemá kolem sebe tak velkou komunitu. To se odráží i na množství dohledatelných rad a postupů na internetu, nebo dostupnosti různých hotových komponent. Právě integrace Bootstrapu byla jedním z dalších důvodů, Nette jej v posledních verzích také podporuje, ovšem nedosahuje tak velkého stupně integrace.

V neposlední řadě také rozhodla podpora Yii frameworku v programovacím nástroji PHPStorm.

5.1.2.2 DRY přístup

DRY, neboli „Don't repeat yourself" přístup říká, že každá část systémových znalostí by měla mít jedno jednoznačné zastoupení. Jedná se o princip vývoje softwaru zaměřený na redukci počtu duplicitních informací všeho druhu. Tento přístup se ovšem netýká pouze zdrojového kódu, vztahuje se i na databázová schémata, testovací plány, dokonce i dokumentaci.

Pokud se totiž (nejen) ve zdrojovém kódu nachází více způsobů vyjádření té samé věci, časem se mohou začít rozcházet. A i kdyby ne, udržovat je všechny aktuální stojí velké vypětí sil a času. Proto pokud chce vývojář flexibilní a udržovatelný systém, měl by se držet DRY přístupu.

Chyby v tomto přístupu se označují jako WET, „We enjoy typing" nebo také „Write everything twice". [23]

5.1.3 Bootstrap

Bootstrap je frontendový CSS a JavaScript framework vytvořený firmou Twitter. Jedná se o sadu komponent, které usnadňují a urychlují vývoj webových stránek. Těmi jsou

[9] 3. nejoblíbenější framework dle http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

tlačítka, modální okna, záložky, tooltipy, formulářové prvky, ikony a další více či méně užívané komponenty.

Největší výhodou Bootstrapu je fakt, že zdarma poskytuje množství nástrojů, které pomáhají vývojáři tvořit flexibilní a responzivní weby stejně jako prvky uživatelského rozhraní. [24]

Mezi další výhody patří:

- Šetří vývojářův čas. Díky hotovým komponentám a layoutům, které se dají snadno upravit, může vývojář soustředit pozornost na více programátorské činnosti, které jsou na pozadí webu.
- Konzistentní vzhled. Používáním komponent z Bootstrapu se nemůže stát, že by se na web dostaly například tlačítka s různým vzhledem, protože tyto sdílí jednotný design a styly.
- Jednoduchost. Každý se základní znalostí HTML a CSS je schopen se velice rychle naučit používat Boostrap.
- Responzivita. Bootstrap má zabudované mechanismy a nástroje, které pomáhají řešit vzhled webu na různých zařízeních.
- Kompatibilita s prohlížeči. Nemůže se tak stát, že by se web špatně zobrazoval na některém z moderních prohlížečů.
- Open Source. Ohromnou výhodou pro velké množství vývojářů je i fakt, že Boostrap je kompletně zdarma.

Bootstrap byl v rámci této práce využit z několika důvodů. Tím nejzásadnějším je fakt, že je plně integrován v Yii frameworku, což činí jeho použití ještě snadnějším. Dalším důvodem je jeho vhodnost pro tento typ webů díky jednotnému vzhledu. A protože byla jednoduchost jedním z požadavků na informační systém, splnil tento účel na 100 %. Díky hotovým komponentám nebylo nutné řešit například kompletní vytváření a stylování formulářových prvků tak, aby systém vypadal kompaktně. Stejně tak dosažení požadované responzivity včetně kompatibility v různých prohlížečích bylo s tímto frameworkem mnohem jednodušší.

5.1.4 PostgreSQL

PostgreSQL je relační databázový systém vyvíjený pod MIT licencí, díky tomu je dostupný zdarma a vývojáři mají k dispozici otevřené zdrojové kódy. Je již 17 let aktivně vyvíjen (poslední verze 9.4.1 uvolněna 5. února 2015) a za tu dobu si získal výbornou pověst díky své spolehlivosti a bezpečnosti. Je možné ho provozovat na všech běžně rozšířených operačních systémech včetně BSD nebo Solaris.

Stoprocentně splňuje podmínky ACID[10], plně podporuje cizí klíče, JOIN operace, pohledy, triggery, uložené procedury, obsahuje nejen většinu SQL92 a SQL99 datových typů jako integer, numeric, varchar, interval nebo timestamp, ale i moderní datové typy jako JSON nebo XML. Výkonnostně nezaostává za srovnatelnými komerčními systémy a v nových verzích je mnohdy předčí. PostgeSQL umožňuje běh procedur napsaných v různých programovacích jazycích – Perl, Python, C nebo speciální PL/pgSQL jazyk vycházející z PL/SQL firmy Oracle.

Předností PostgreSQL je také jeho rozšiřitelnost. Může tak být rozšiřován o nové datové typy, funkce, operátory, agregační funkce nebo procedurální jazyky. Díky tomu vzniklo několik různých rozšíření, například PostGIS - podpora pro geografické informační systémy. [25] [26]

Výhody:

- Je zdarma, stále probíhá aktivní vývoj.
- Různé programovací jazyky pro uložené procedury.
- Anonymní procedury. Hodí se v případě používání update skriptů na databáze.
- Transakční zpracování. Pokud dojde během transakce k chybě, další příkazy se již neprovedou.
- Podpora různých datových typů, včetně uživatelsky definovaných.
- Velmi kvalitní dokumentace s českým překladem FAQ.

Nevýhody:

- PHP je spíše provázané s MySQL – více tutoriálů a příkladů.
- S tím souvisí i druhá nevýhoda – minimum webových hostingů na českém trhu, které podporují PostgreSQL.

[10] Atomic, Consistent, Isolated, Durable

- Horší podpůrné nástroje. pgAdmin nedosahuje kvalit MySQL Workbench, oblíbený Adminer má při použití s PostgreSQL více chyb.

PostgreSQL bylo pro účely této práce vybráno z především z důvodu téměř dvou let zkušeností díky využívání v zaměstnání. S tím souvisí také značné využití uložených procedur v informačním systému. Nedostatek v případě nedostupnosti kvalitních nástrojů pro práci s PostgreSQL vyvažuje fakt, že kromě příkazu psql pro příkazovou řádku a nástroje pro vygenerování ER diagramu nebyly jiné nástroje vyžadovány.

5.1.5 LESS

Udržet v rozsáhlém webovém projektu snadno čitelné a udržovatelné CSS styly stojí velkou míru úsilí, které je s využitím vhodných nástrojů možné směrovat na důležitější činnosti. Tak, jako jednoduchému návrhu vzhledu pomáhá Bootstrap, tak LESS pomáhá udržovat jej a dále vyvíjet.

LESS, neboli Leaner CSS, je knihovna, která zavádí do CSS stylů vlastnosti, které jsou známé z jiných programovacích jazyků, jakými jsou proměnné nebo funkce.

```
@import „mixiny.css“;

@module-icon-shadow: 0px 0px 10px 0px rgba(0,0,0,0.75);

/** loadovací mixin */
.loading() {
    position: relative;

    &:after {
        display: none;
        content: '';
        position: absolute;
        background-color: rgba(255, 255, 255, 0.5);
        background-image: url("../images/ajax-loader.gif");
        background-repeat: no-repeat;
        background-position: center center;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        z-index: 999;
    }

    &.loading:after {
        display: block;
    }
}
```

Ukázka kódu 4: Ukázka funkcí LESSu, zdroj: vlastní

Na konkrétním příkladu z vytvářeného informačního systému je vidět definování „proměnné“ *@module-icon-shadow*, funkce (v LESS nazývané mixin) *.loading* a vnořování pravidel.

Toto ovšem nejsou jediné přidané možnosti oproti CSS. Mezi další přednosti patří vkládání souborů pomocí *@import „mixiny.css“*, řádkové komentáře, výrazy nebo struktury.

Takový zápis stylů ovšem prohlížeč nedokáže sám interpretovat, proto je nutné LESS „přeložit“ na běžné CSS. To lze provést pomocí utility v příkazovém řádku, na serveru jej konvertovat skriptem nebo jej vložit přímo do stránky a o jeho zpracování se pak stará prohlížeč pomocí JavaScriptového nástroje less.js.

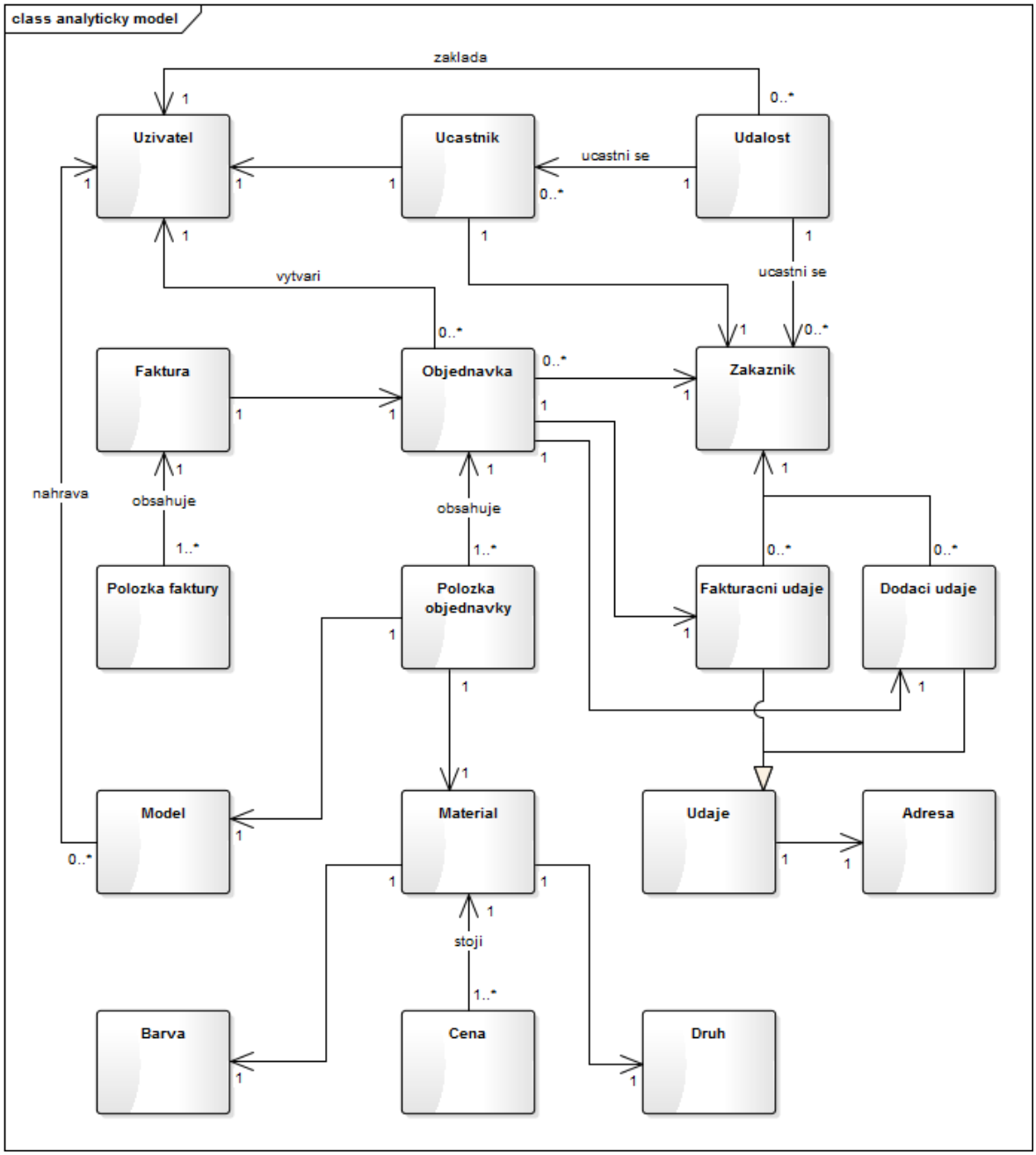
I přes značné výhody má LESS i jednu drobnou nevýhodu. Pokud na projektu spolupracuje více vývojářů, musí každý pracovat pouze na LESS souboru. Pokud by totiž někdo upravil výsledný CSS soubor, s další konverzí by o tyto úpravy přišel. [27]

Důvodem využití preprocesoru LESS bylo především zjednodušení zápisu CSS stylů. Díky vnořování pravidel je možné snadno přetěžovat už deklarovaná (třeba v rámci Bootstrapu) pravidla a nepřijít přitom o přehlednost, čehož není s běžnými CSS snadné dosáhnout. Díky integraci LESS pluginu v PHPStormu je jeho použití ještě jednodušší a odpadá tak problém s dodatečnou konverzí na CSS.

5.2 Návrh datové struktury

Analýza požadavků na informační systém poskytla podklady pro tvorbu analytického modelu tříd. V něm jsou vzájemně propojeny třídy, které tvoří základní stavební prvky vyvíjeného systému. Pro tvorbu modelů tříd byl využit program Enterprise Architect ve verzi 11.

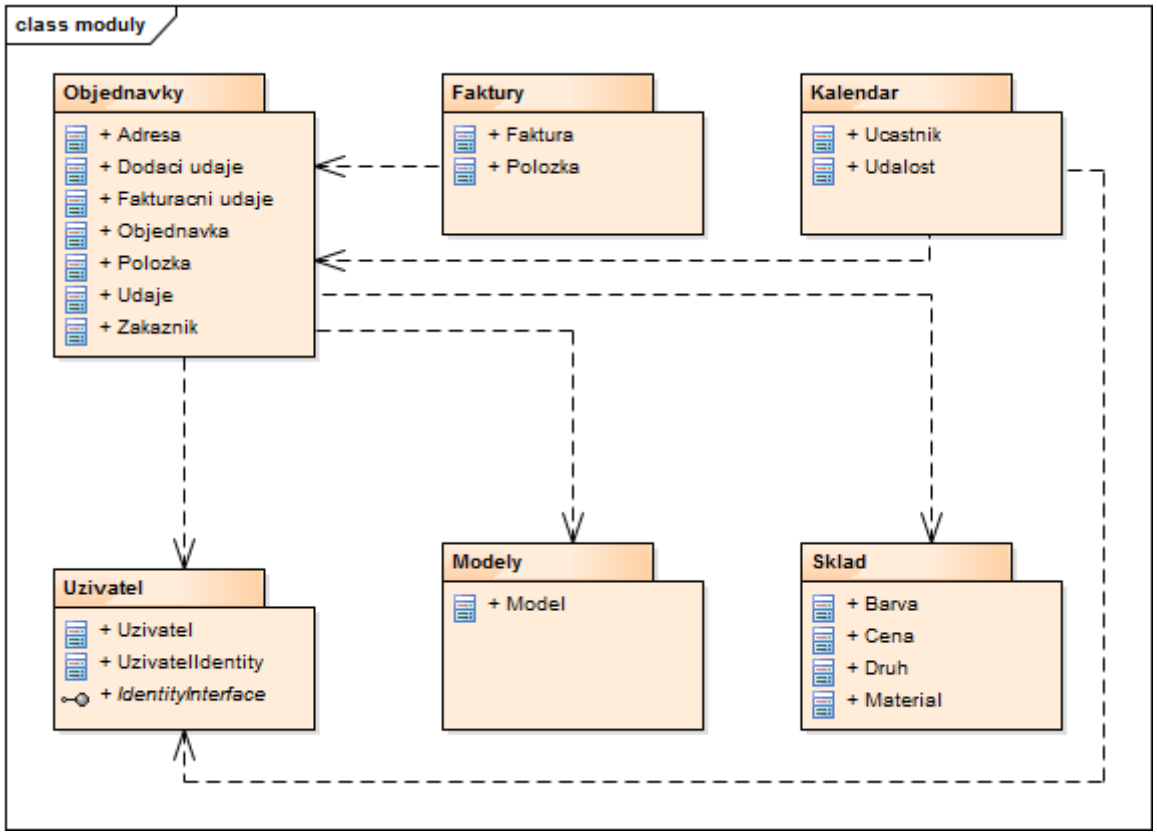
5.2.1 Analytický model tříd



Obrázek 12: Analytický model tříd, zdroj: vlastní

Kvůli velikosti výsledného modelu byly vynechány atributy a metody jednotlivých tříd. Atributy jsou součástí návrhového modelu tříd, viz následující kapitola.

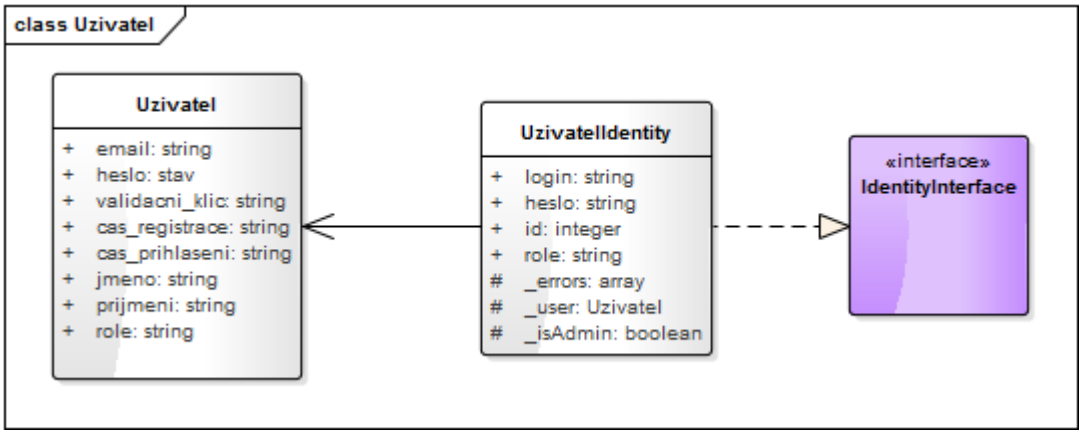
5.2.2 Návrhový model tříd



Obrázek 13: Návrhový model tříd rozdělený do balíčků, zdroj: vlastní

Třídy byly z důvodu přehlednosti uspořádány do balíčků tak, jak vystupují v jednotlivých modulech. V následujících kapitolách jsou v krátkosti popsány některé balíčky a třídy se svými atributy.

5.2.2.1 Modul „Uzivatel"



Obrázek 14: Class diagram modulu Uzivatel, zdroj: vlastní

Třída „Uzivatel"

Třída *Uzivatel* odpovídá uživatelům v systému. Atribut *email* vystupuje jako uživatelské jméno v systému, *role* je staticky přidělována v administraci a při zakládání nového uživatele mu může být přidělena stejná nebo nižší role podle definované hierarchie a role přihlášeného uživatele. Atribut *validacni_klic* slouží k ověření uživatele při obnově zapomenutého hesla.

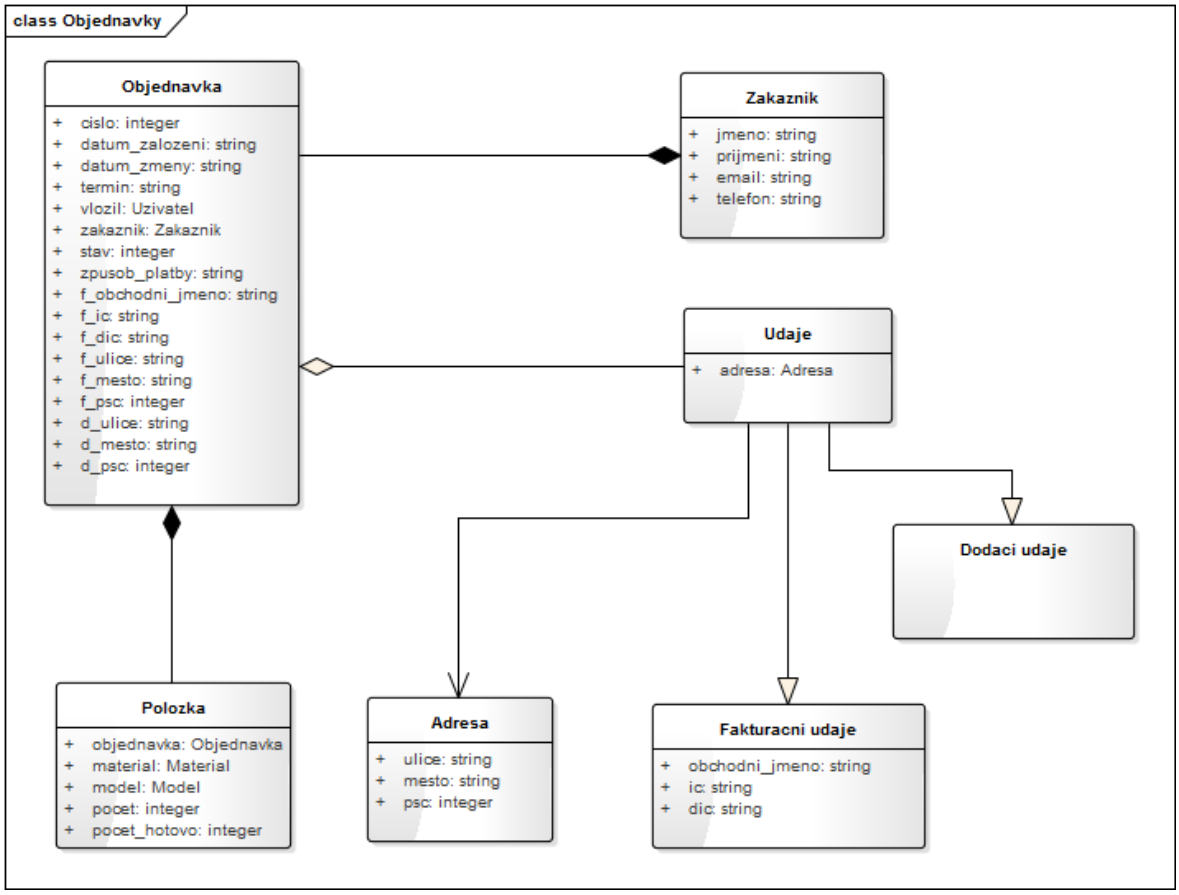
Třída „UzivatelIdentity"

Tato třída představuje přihlášeného uživatele v systému. Drží si základní identifikátory jako id, login (email) nebo roli. Pomocí této třídy je možné ověřovat oprávnění při práci s entitami jako objednávka nebo model.

Interface „IdentityInterface"

Definuje metody, které jsou frameworkem vyžadovány pro práci s identitou uživatele například při přihlášení nebo ověření pomocí validačního klíče/tokenu. Konkrétní implementace metod je na vývojáři, framework pouze vyžaduje, aby návratovou hodnotou metod byla třída, která implementuje *IdentityInterface*.

5.2.2.2 Modul „Objednavky“



Obrázek 15: Class diagram modulu Objednavky, zdroj: vlastní

Třída „Objednavka“

Objekt představuje založené objednávky, které si drží referenci na zákazníka, který poptává zboží. Každá objednávka má mimo jiné přidělené unikátní číslo, má termín splnění a také stav, který se mění automaticky v průběhu zpracovávání objednávky v systému, nebo je možné jej změnit ručně.

Do objednávky se také propisují dodací a fakturační údaje vybrané při založení objednávky. Toto řešení bylo zvoleno z důvodu jednoduchosti, protože není nutné udržovat u údajů jejich časovou platnost.

Třída „Polozka“

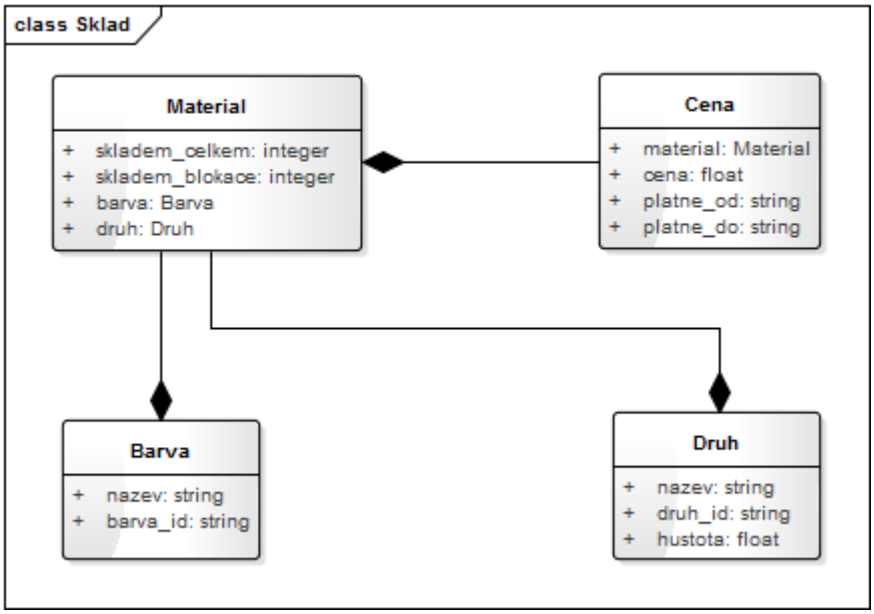
Jednotlivé položky objednávky, které odkazují na model a materiál, ze kterého bude model vyroben. Dalšími atributy jsou již pouze počet kusů dané položky a počet hotových kusů. Tato informace slouží k udržování přehledu o nedokončených objednávkách. Ve chvíli, kdy je roven počet kusů a počet hotových kusů každé položky

objednávky, je objednávka automaticky označena jako „Hotová“ a v systému se vygeneruje faktura.

Třídy „FakturacniUdaje“ a „DodaciUdaje“

Třídy, které mají společného předka – „*Udaje*“. Ta si drží referenci na adresu, protože je společná pro obě třídy. Rozdělení na potomky bylo zvoleno z důvodu odlišných vlastností nad rámec adresy. Dodací údaje v tuto chvíli obsahují pouze adresu, ale samostatná třída byla zvolena pro možnost budoucího rozšíření například o kontaktní telefon nebo poznámku.

5.2.2.3 Modul „Sklad“



Obrázek 16: Class diagram modulu Sklad, zdroj: vlastní

Třída „Material“

Materiál je další třídou (vedle Modelu), jejíž referenci si drží položka objednávky. Na základě hustoty materiálu a objemu modelu je vypočítána cena za vytištění jednoho kusu položky. Aby nebylo možné zakládat objednávky na modely z nedostupného materiálu, drží si tato třída informaci o celkové skladové dostupnosti a objemu blokovaného materiálu.

Třídy „Barva“, „Druh“, „Cena“

Pomocné třídy, které obsahují názvy entit, identifikátory případně další vlastnosti. Třída cena je uzpůsobena pro verzování, proto má atributy určující její platnost v časovém intervalu. Položka objednávky (materiál) tak ukazuje vždy na platnou cenu z doby, kdy

byla objednávka potvrzena. Této vlastnosti je využito i pro sledování vývoje ceny materiálu za určitý čas.

5.3 Moduly informačního systému

Moduly, které musí, a bude informační systém obsahovat, vycházejí z analýzy firmy a firemních procesů, požadavků na informační systém a konzultací se zástupci firmy. Jako podklad pro tvorbu modulů sloužil návrhový model tříd.

5.3.1 Uživatelský modul

Uživatelský modul představuje základní komponentu aplikace, bez které by informační systém vůbec nefungoval. Modul zajišťuje vlastní autentifikaci uživatelů do systému, jejich komplexní správu a přidělování rolí. Protože se jedná o uzavřený systém, uživatele přidává do systému administrátor, který jako jediný má přístup do celého modulu. Pro ostatní uživatele je přístupná pouze část – „Můj profil“, ve které si mohou měnit své údaje a heslo.

Oprávnění jsou v současné době přidělovány uživatelům staticky na základě rolí, protože systém byl vyvíjen pro potřeby malého počtu uživatelů, kteří mají v celém systému de facto stejné pravomoci. Ovšem s dalším rozšiřováním systému a růstem počtu zaměstnanců, kteří budou se systémem pracovat, se počítá s vytvořením správy rolí a dynamického řízení přístupu do jednotlivých modulů a k jednotlivým akcím na základě přidělených rolí.

5.3.2 Objednávky

Objednávkový modul reflektuje nejdůležitější firemní proces. V rámci modulu je možné přidávat objednávky, upravovat je ve smyslu změny dodacích nebo fakturačních údajů, změny položek a jejich počtu nebo změny stavu. Veškeré úpravy objednávky se řídí definovanými pravidly, takže například objednávku, která nemá vyplněné některé údaje, není možné vyfakturovat. Tato pravidla jsou definována staticky a týkají se všech uživatelských rolí, které v systému vystupují.

V přehledu objednávek se nachází filtr, který umožňuje jednodušší dohledávání konkrétních objednávek. Každou objednávku je možné zobrazit v detailním přehledu, ze kterého je možné přímo editovat některé údaje, pokud to stav objednávky umožňuje.

Smazání objednávky je realizováno přepnutím do stavu „Stornováno systémem“. To odpovídá zásadě, že v informačním systému by se neměla žádná data mazat.

Evidovat objednávky bez zákazníků by nemělo smysl, proto je součástí tohoto modulu i kompletní evidence zákazníků. Jako v případě objednávek, i zde je možnost kompletního výpisu podle filtru, přidávání a editace. V detailu každého zákazníka je možné zobrazit a přímo editovat jeho dodací nebo fakturační údaje, součástí je i výpis všech objednávek zákazníka, které umožňují proklik na detail.

Modul je napojen na další části systému, kterými jsou správa zákazníků (spadá do tohoto modulu), evidence skladové dostupnosti (kapitola 5.3.3) a 3D modely (kapitola 5.3.4). Navíc je propojen s hlavní obrazovkou, kde se zobrazují nové nevyřízené objednávky, objednávky, na kterých se aktuálně pracuje a také objednávky, které jsou po termínu splnění.

5.3.3 Skladové zásoby

Modul skladových zásob slouží pro přehled aktuálně dostupného materiálu, přidávání nových materiálů nebo jejich doplňování na sklad. Každý materiál má definovánu cenu s platností od – do, barvu, množství skladem, blokaci, ale především také hustotu, ze které se vypočítává konečná cena modelu na základě jeho objemu. Díky propojení objednávek na skladový modul není možné zakládat objednávky z nedostupného materiálu.

Detail jednotlivých materiálů obsahuje pouze graf za poslední rok + aktuální měsíc. Ten zobrazuje vývoj průměrné ceny materiálu a počet objednávek v jednotlivých měsících. Nepředpokládá se častá fluktuace cen, proto bylo zvoleno rozdělení po měsících, které je naprosto dostačující.

5.3.4 3D modely

Další modul, který je přímo propojen s objednávkami, kdy 3D modely uložené v systému zastupují funkci položek objednávky. Modul navíc slouží jako jednotné úložiště modelů, takže se nejedná o pouhou evidenci, ale umožňuje také stahování fyzických souborů.

Důležitou vlastností modelů je jejich objem. Jak bylo zmíněno v předchozí kapitole, na základě objemu modelu, hustoty materiálu a jeho ceny je vypočítána výsledná cena.

V drtivém množství případů jsou do systému nahrávány binární STL soubory[11] , proto byla pro systém vytvořena knihovna, která soubor rozparsuje a z vrcholů vypočítá objem modelu. V případě, že se během této operace vyskytne chyba (je nahrán soubor jiného typu, chyba v datové struktuře), je uživatel vyzván k zadání objemu v samostatném formuláři.

Součástí detailu modelu je také jeho 3D zobrazení. Zobrazení v tomto případě spoléhá na volně dostupnou knihovnu ThingiView[12], která je napsaná v jazyce JavaScript. Ta za pomocí technologií HTML5 a WebGL dokáže vykreslit 3D objekty do HTML stránky. Knihovna byla lehce upravena a nyní umožňuje pořizovat snímky zobrazovaného modelu, například pro potřeby budoucího katalogu nebo eshopu.

5.3.5 Kalendář

Jednoduchý modul, který je tvořen především základní obrazovkou se zobrazením celého měsíce. Modul slouží k plánování schůzek nebo jednání, zapisování poznámek a úkolů. Je možné zadávat i vícedenní události.

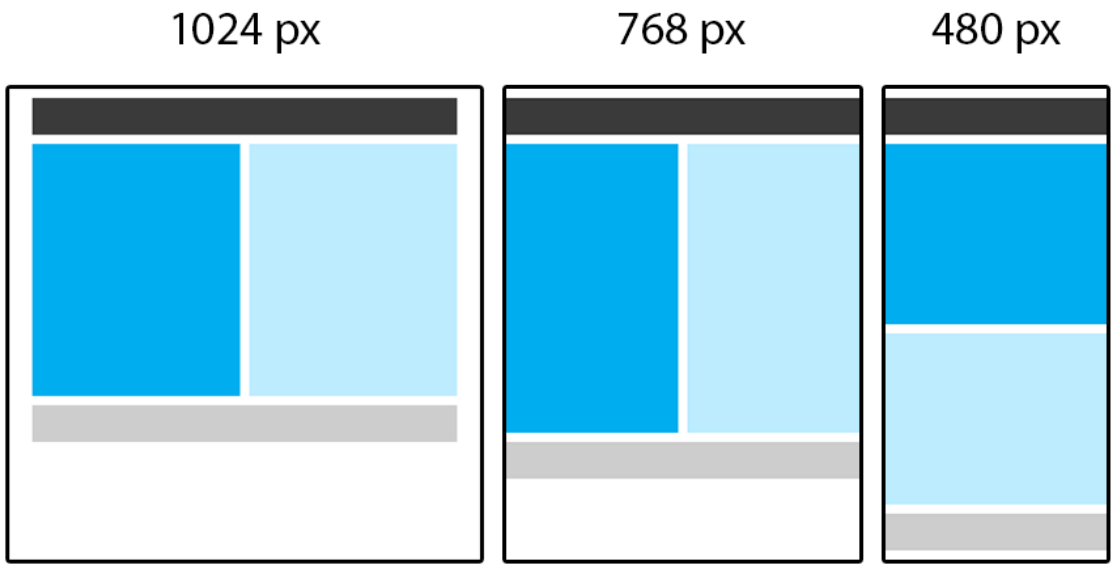
Tento modul je také propojen s hlavní obrazovkou, kde jsou zobrazovány poslední proběhlé a probíhající události v jedné tabulce a v další pak nejbližší nadcházející události.

Ke každé události je možné přidat další účastníky. Všem účastníkům se pak akce zobrazuje na hlavní obrazovce a v kalendáři.

5.4 Návrh uživatelského prostředí

Uživatelské prostředí informačního systému se z velké části skládá z formulářů a rozsáhlých tabulek. Obojí je v případě mobilního zobrazení nutné zobrazit v takové podobě, aby nezabíraly většinu průhledu, ale stále poskytovaly dostatek informací nebo umožňovaly snadnou práci. K tomuto účelu je využíváno přeskupování prvků v rámci stránky.

[11] Stereolitografie, soubor popisuje 3D model pomocí vrcholů a normál v kartézském systému souřadnic
[12] https://github.com/tbuser/thingiview.js



Obrázek 17: Základní návrh přeskupování stránky, zdroj: vlastní

5.4.1 Menu

Menu jako základní ovládací prvek celého systému by mělo být pro uživatele jednoduché, přehledné a snadno ovladatelné.

Pro účely informačního systému bylo menu vytvořeno dvakrát. První, desktopové, je zobrazováno na větších zařízeních (průhledech), zatímco druhé je skryté. Při zmenšení stránky pod určitou úroveň nebo zobrazení na mobilním telefonu se desktopové menu skryje a zobrazí se tlačítko pro ovládání mobilního menu.

Tím je zajištěno, že se bude menu vždy snadno ovládat a nebude zabírat místo, kterého je na mobilním telefonu nedostatek, a na jeho místě tak mohou být jiné prvky systému. Například více tabulek nebo formulářových prvků.

Pro snadnou navigaci byly navíc vytvořeny barevné ikony představující jednotlivé moduly. Předpokládá se, že svou barevností rozbijí jinak jednotvárný systém, a navíc zjednoduší navigaci tím, že nebude nutné číst nebo hledat popisky. Uživatel si snadno si zapamatuje, že například uživatelský modul má zelenou ikonu, a dále se orientuje již právě podle barvy.

5.4.2 Formuláře

Velká část formulářů byla rozdělena na dvě části (sloupce) tak, aby jím byly ideálně vyplněny celé stránky a nebylo nutné zbytečně rolovat stránkou. Výjimkou jsou filtry, které jsou rozdělené na tři sloupce, a některé menší formuláře, které naopak tvoří pouze jeden sloupec (například změna hesla).

Díky využití CSS frameworku Bootstrap nebylo nutné vymýšlet vlastní způsoby, jak by se měly formuláře chovat při používání různých rozlišení. Bootstrap tuto funkcionalitu řeší a jediné, co vývojář musí udělat, je správně rozvrhnout umístění jednotlivých formulářových polí tak, aby se na menším zobrazení logicky uspořádaly. Rozdělením formulářů na sloupce byla související pole spojena do skupin, aby zůstávaly pohromadě.

5.4.3 Responzivní tabulky

Zobrazení rozsáhlých tabulek na webu tak, aby odpovídaly responzivnímu designu, je možné řešit různými způsoby. Nejčastější jsou ale následující tři.

5.4.3.1 Vypouštění sloupců

Řešení velmi jednoduché a účelné. Velkou výhodou je jednoduchost CSS stylů a také fakt, že tabulka zůstává v původní podobě a tak není problém s řazením dat podle jednotlivých sloupců.

Naopak nevýhodou je právě ono vypouštění sloupců. Je nutné dobře rozmyslet, která data nejsou tolik důležitá a je tudíž možné je uživateli nezobrazit.

5.4.3.2 Horizontální scrollování

Další velmi jednoduché řešení, jak se vypořádat s rozsáhlými tabulkami na malém rozlišení mobilních telefonů. Výhody jsou podobné jako v případě vypuštění sloupců – jednoduchý zápis a tabulka v původní podobě.

Nevýhodou je, že prvek, kterého se responzivní design snaží co nejvíce (resp. úplně) zbavit – horizontální scrollování – v tomto případě zůstává. Nicméně je to pouze malý ústupek použitelnosti a webová stránka stále zůstává v horizontálním směru statická.

5.4.3.3 Přeskupení tabulky

Přístup, který nejvíce odpovídá responzivnímu designu. Celá tabulka se přeskupí tak, že se z jednotlivých řádků vytvoří nové tabulky. Záhlaví původní tabulky se přesune do prvního sloupce nové tabulky, v druhém jsou vlastní hodnoty odpovídající jednotlivým řádkům původní tabulky.

Výhodou tohoto řešení je následování principů responzivního designu a vypuštění nutnosti horizontálně scrollovat (ať již tabulkou nebo celou stránkou).

Nevýhodou je složitější zápis CSS stylů, omezená funkcionalita v některých starších prohlížečích (typicky IE8 a nižší), a také menší přehlednost, která vynikne ve spojení s rozsáhlými tabulkami. Menší přehlednost je možné řešit kombinací přeskupení tabulky a vypouštění sloupců, ale v takovém případě je lepší se zamyslet, zda je skutečně nutné mít tak rozsáhlou tabulku a zda by nebylo lepší data strukturovat jinak.

Existuje ještě několik dalších řešení responzivních tabulek, které ovšem nejsou tak časté. Jedná se například o kombinaci přeskupení tabulky a horizontálního scrollování nebo nahrazení tabulky grafem. To lze ovšem použít pouze pro tabulky, jejichž hodnoty umožňují transformaci do grafu.

Pro účely této práce byl vybrán druhý přístup, tedy horizontální scrollování. Rozhodla především jednoduchost vlastního řešení, ale také konzultace se zástupci firmy, kdy se jim toto řešení zdálo elegantnější a především lépe použitelné.

5.5 Návrh wireframů

Fáze návrhu wireframů následuje po celkové analýze a návrhu systém uživatelského prostředí. Během této fáze se hrubě rozvrhnou ovládací prvky, sjednotí uživatelské rozhraní a umístí na svá místa všechny důležité prvky systému. Díky tomu je možné získat náhled nad jednotlivými obrazovkami informačního systému ještě před započetím vlastního programování.

Pro tvorbu wireframů byl použit nástroj Balsamiq Mockups[13], který umožňuje jednoduchou tvorbu graficky přívětivých návrhů. Jeho výhodou je také snadný export všech otevřených obrazovek do formátu PNG nebo do jednoho PDF souboru, ve kterém fungují i odkazy na jednotlivé obrazovky.

[13] https://balsamiq.com

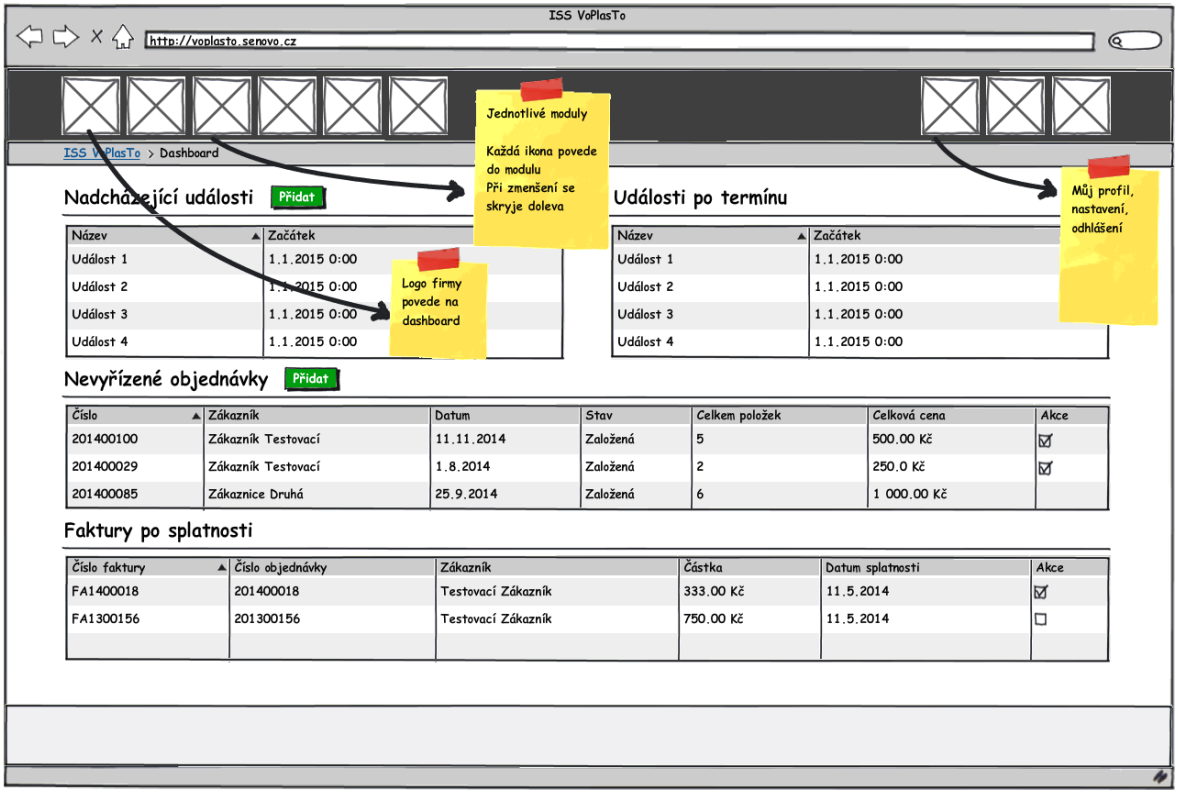
Wireframy byly vytvořeny ve dvou rozměrech, a to pro standardní zobrazení na stolním počítači a poté pro zobrazení na mobilním telefonu. Celkem bylo vytvořeno šest obrazovek ve standardním zobrazení a tři obrazovky pro mobilní telefon. V samostatných kapitolách jsou popsány dvě nejdůležitější části informačního systému – hlavní obrazovka a formulář pro přidání nové objednávky.

Společným prvkem uživatelského rozhraní je hlavička a patička stránky. Patička stránky je pouze jakýmsi vizuálním ukončením stránky a obsahuje pouze email na podporu, proto není potřeba ji popisovat. Hlavička je po vzhledové stránce totožná pro desktopové i mobilní zobrazení, obsah se ovšem přizpůsobuje šířce průhledu. V horní, tmavé, části obsahuje ovládací prvky menu, ve spodní světlé části pak drobečkovou navigaci a uživatelské jméno právě přihlášeného uživatele. Při prohlížení na dostatečně širokém zařízení je menu zobrazené v podobě velkých ikon a rozdělené na dvě části. Levá obsahuje odkazy na hlavní stránku a jednotlivé moduly, pravá část obsahuje systémové položky. Po zmenšení stránky pod určitou hranici (zde 768 px) se horní menu a uživatelské jméno skryjí a zobrazí se prvky uzpůsobené pro ovládání na tabletech a mobilních telefonech. V tomto případě se jedná pouze o jediné tlačítko, které zobrazí nebo skryje mobilní menu.

Mobilní menu obsahuje všechny odkazy, které se nachází v desktopové variantě, nejsou ale rozděleny na části a jsou seřazeny pod sebou. Položky menu jsou tvořeny textovými názvy modulů včetně ikon, ty jsou pouze v menším rozměru, aby odpovídaly velikosti odkazů. Menu po otevření odsune celou stránku za pravý okraj zařízení, což vytváří zajímavý grafický efekt.

5.5.1 Hlavní obrazovka

Hlavní obrazovka slouží pro rychlý přehled nad celým informačním systémem. Pokud si ji uživatel otevře, měl by se dozvědět co nejvíce podstatných informací, aniž by musel navštívit jednotlivé moduly. Proto byly pro zobrazení vybrány nadcházející události, nové nezpracované objednávky, objednávky po termínu dokončení a také faktury po splatnosti.



Obrázek 18: Wireframe pro hlavní obrazovku IS, zdroj: vlastní

Ve výpisech se nezobrazují všechna data vyhovující výše zmíněnému nastavení, zobrazeno je vždy pouze deset nejaktuálnějších[14] položek, bez stránkování. Události jsou řazeny podle data začátku od nejbližší, objednávky a faktury podle data založení od nejstarších. Tedy tak, aby jako první byly vždy položky, které je potřeba co nejdříve vyřídit.

Ve výpisech událostí jsou vždy zobrazeny název a datum včetně odkazů na rychlé akce. Ty jsou specifické pro každý typ události, u úkolů je možné například potvrdit jeho splnění a podobně. Název události umožňuje proklik na její detail.

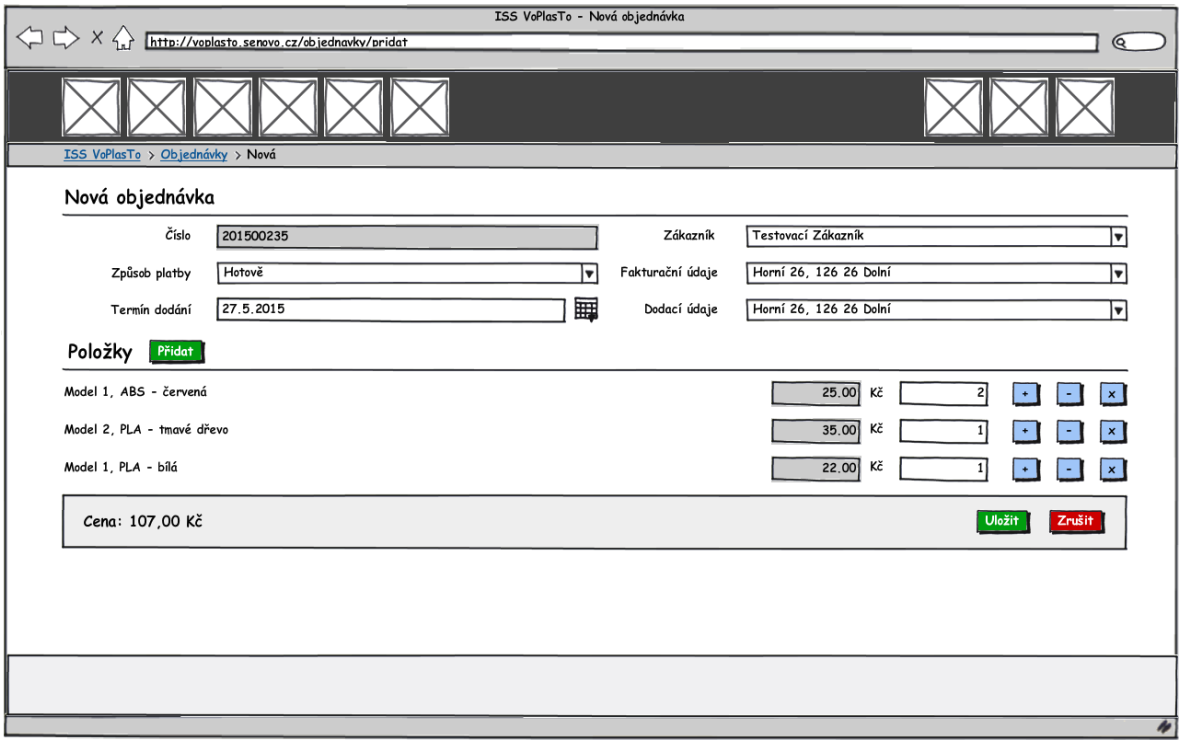
Výpisy objednávek a faktur obsahují údaje, které jsou pro uživatele důležité, jako například číslo objednávky nebo faktury, jméno zákazníka, datum, částku a další. Součástí výpisu jsou u jednotlivých položek také rychlé akce, které umožňují zobrazit detail dané položky, případně přejít na její editaci. Stejně jako v případě událostí, i čísla objednávek, čísla faktur a jména zákazníků vedou na jejich detaily, aby nebylo nutné je dohledávat v samostatných výpisech.

[14] Ve smyslu, že brzy nastanou nebo je potřeba je co nejdříve zpracovat.

Součástí hlavní obrazovky jsou také tlačítka, která slouží jako rychlé odkazy na přidání objednávek a událostí a nacházejí se u souvisejících nadpisů.

5.5.2 Nová objednávka

Jak již bylo zmíněno v úvodu kapitoly 5.5, formulář pro vytvoření nové objednávky je vedle samotného výpisu objednávek asi tou nejdůležitější částí celého informačního systému. Původní návrh obsahoval formulář rozdělený vertikálně, tedy informace o objednávce v levé části, položky v pravé. Z důvodu přehlednosti bylo později od tohoto řešené upuštěno a formulář byl navržen s horizontálním rozdělením, viz Obrázek 19 níže.



Obrázek 19: Wireframe formuláře pro novou objednávku, zdroj: vlastní

Číslo objednávky je jediné needitovatelné pole ze základních údajů objednávky. Jeho zobrazení je pouze informativní a aktuální je generováno vždy s uložením celé objednávky, aby nedocházelo ke konfliktům, pokud by pracovalo s formulářem více uživatelů najednou. Ostatní pole jsou už editovatelná, fakturační a dodací údaje jsou závislé na výběru zákazníka, termín dodání je zadáván pomocí komponenty pro výběr data a není možné vybrat dřívější, než je následující den.

Položky do objednávky uživatel vybírá pomocí modálního okna otevíraného tlačítkem s popiskem „Přidat“. Součástí výpisu modelů je i filtr na název modelu, uživatel proto

nemusí výpis procházet celý. U každé položky výpisu je přítomna i volba materiálu, ze kterého bude model vytištěn. Toto řešení bylo zvoleno záměrně, neboť udržovat ke každému modelu položky v podobě materiálů by bylo příliš složité. Po přidání položky k objednávce se pod formulářem objeví řádek s názvem modelu, vybraným materiálem a barvou v levé části, cenou, počtem kusů a ovládacími tlačítky v části pravé. Ovládací tlačítka slouží k navyšování nebo snižování počtu kusů položky v objednávce nebo k jejímu úplnému odebrání. Ve spodní části obrazovky se přepočítává aktuální cena za objednávku.

Jedním z požadavků na tuto obrazovku byla možnost přidat objednávku i bez vybraného zákazníka. Takové zadání ovšem postrádalo smysl a byl proto ujednán kompromis. Z objednávkového formuláře může uživatel „rychle“ založit nového zákazníka pouze s kontaktními údaji. Objednávku bez fakturačních a dodacích údajů už je možné uložit, na jejich nepřítomnost je uživatel upozorněn a po uložení objednávky se v jejím detailu zobrazuje další upozornění. Takovou objednávku navíc není možné vyfakturovat, dokud uživatel údaje nedoplní.

Další vytvořené wireframy se nacházejí v příloze.

5.6 Vizualizace prostředí

Po schválení návrhu uživatelského rozhraní a vytvořených wireframů se postupně přešlo k vizualizaci vybraných obrazovek informačního systému. Pro tuto fázi ještě není nutná veškerá funkcionalita, proto byly vytvořeny pouze jednotlivé obrazovky s rozvržením požadovaných prvků informačního systému. Aby bylo možné posoudit vzhled, byly aktivní prvky ručně naplněny smyšlenými testovacími daty, což umožnilo alespoň částečné vyzkoušení systému.

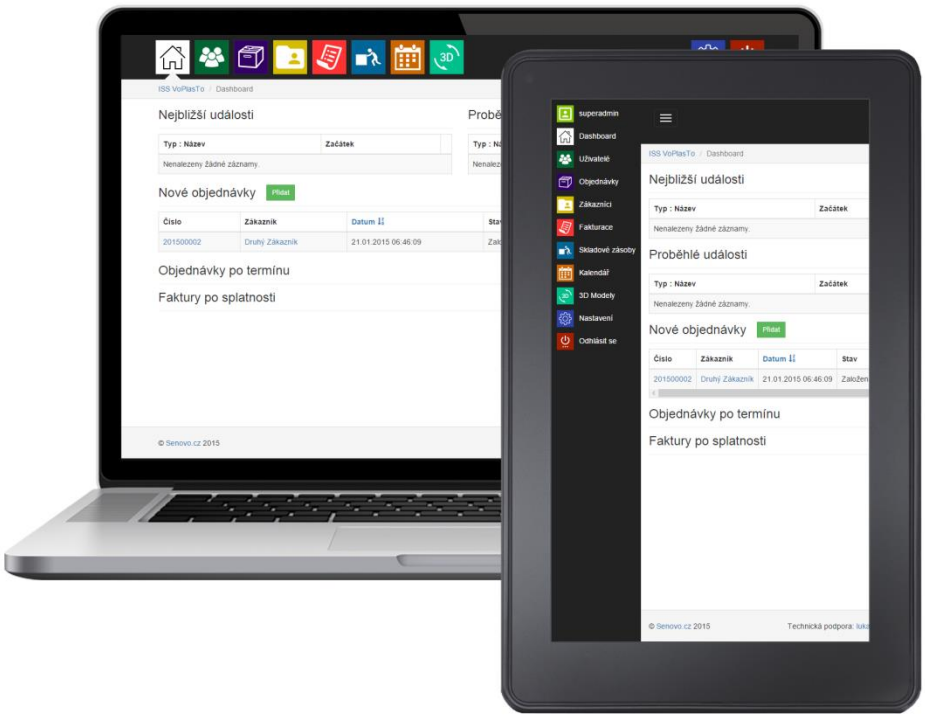
Obrázky s vizualizacemi byly vytvořeny pomocí nástroje dostupného na webu http://deviceponsive.com.

Tato fáze ale především posloužila k rychlému otestování uživatelského rozhraní na různých zařízeních. K testování byla použita jak fyzická zařízení, tak i softwarové nástroje dostupné z webu nebo zabudované přímo v prohlížečích internetu. Tyto nástroje, například Chrome Developer Tools nebo Firebug, umožňují simulaci různých

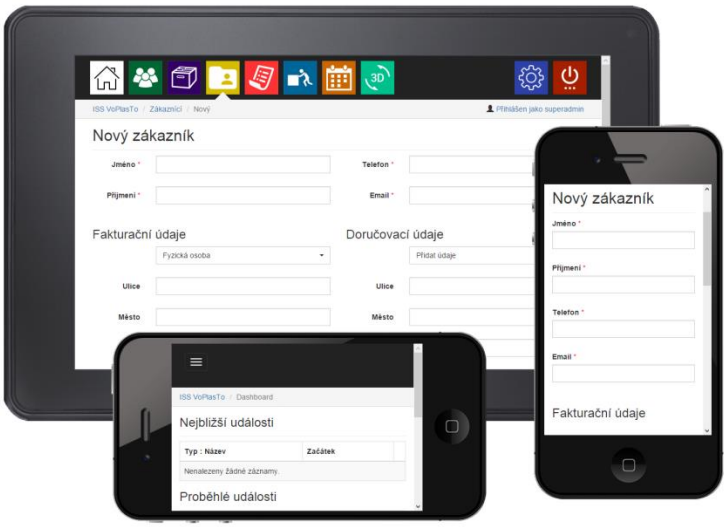
zařízení buď výběrem konkrétního, nebo vlastním nastavením rozlišení a *device-pixel-ratio*.

Uživatelské prostředí bylo navrženo s ohledem na všechna zařízení, která jsou ve firmě využívána k práci. Na vizualizacích níže je možné vidět, že s informačním systémem bude možné jednoduše pracovat jak na počítačích, tak i na mobilních telefonech. Uživatelské prostředí vychází z návrhu v kapitole 5.4, a tak se na menších zařízeních (nejen) formulářové prvky místo vedle sebe zarovnávají pod sebe. Díky tomu nedochází ke skrývání podstatných informací za svislé hrany obrazovky a jsou tak jednoduše dostupné vertikálním posunutím stránky.

Největší změny se odehrály v navigaci, která opět vychází z návrhu prostředí. Protože by na malých zařízeních zabíralo velkou část obrazovky, bylo navigační menu přeskupeno a skryto za levou hranu obrazovky. K vyvolání menu v tomto případě slouží tlačítko v navigační liště, které zobrazí menu a celou stránku skryje za pravý okraj obrazovky.



Obrázek 20: Vizualizace systému na notebooku a tabletu, zdroj: vlastní



Obrázek 21: Vizualizace systému na tabletu a mobilním telefonu, zdroj: vlastní

5.7 Implementace

Po hotových vizualizacích se postupně přešlo do fáze implementace serverové části systému. V této kapitole budou přiblíženy některé praktické úkoly nebo problémy, které bylo nutné při vývoji řešit.

5.7.1 Struktura aplikace

Struktura aplikace vychází z uspořádání basic aplikace Yii Frameworku, došlo pouze k rozšíření o složky s komponentami, moduly nebo tématy. Struktura s nejdůležitějšími položkami vypadá následovně:

```
inter_is/                   base path aplikace
    composer.json           konfigurační soubor Composeru
    config/                 složka s konfiguračními soubory
        console.php         konfigurace konzolové části aplikace
        web.php             konfigurace webové části aplikace
    commands/               obsahuje třídy pro práci s konzolovou částí aplikace
    components/             do této složky patří ručně vyrobené komponenty
    controllers/            obsahuje základní controllery
    models/                 obsahuje modely
    modules/                obsahuje jednotlivé moduly aplikace
    runtime/                běhová složka webu, obsahuje například logy nebo cache
    themes/                 složka s tématy pro různé stylování aplikace
    vendor/                 složka se soubory frameworku a jiných knihoven
    views/                  složka se šablonami
    web/                    webroot aplikace, obsahuje soubory dostupné z webu
        assets/             do této složky publikuje AssetManager CSS a JS soubory
        index.php           vstupní skript aplikace
    yii                     spouštěcí skript konzolové aplikace
```

Ukázka kódu 5: Sktruktura aplikace, zdroj: vlastní

Další přidané složky už nejsou nutné pro běh aplikace, ale byly přidány pro usnadnění vývoje. Mezi tyto složky patří *db* s databázovými skripty a *dbmodels* s modelem databázového schématu.

5.7.2 Formuláře

Téměř polovina vlastní funkcionality aplikace je tvořena formuláři. Aby aplikace dokázala data správně zpracovat, je nutné vstupy od uživatele řádně zvalidovat. K tomuto procesu slouží metoda *validate()* z třídy Model, kterou každý formulář dědí. Při zavolání validace metoda prochází jednotlivá pravidla definovaná v metodě *rules()* a aplikuje je na data z formuláře.

```
public function rules()
{
    return [
        [
            ['ulice', 'mesto', 'psc'],
            'required',
            'when' => function($model) {
                return $model->dodaci_udaje_pk == null;
            },
            'whenClient' => "function (attribute, value) {
                return $('#dodaciudaje-dodaci_udaje_pk').val() == '';
            }"
        ],
        [['dodaci_udaje_pk', 'adresa_pk', 'zakaznik_pk', 'stejne'], 'safe'],
        ['psc', 'match', 'pattern' => '/^\d{5}$/i']
    ];
}
```

Ukázka kódu 6: Metoda rules() s definicí validačních pravidel, zdroj: vlastní

V ukázce jsou definována validační pravidla pro uložení dodacích údajů. Validátor *required* udává, že atributy z výčtu musejí být vyplněny. Díky prvku *when* a *whenClient* je možné určit podmínky validace jak na straně serveru, tak na straně klienta.

Toto je ukázka jednodušších validátorů, v případě nutnosti složitější validace je vhodné vytvořit samostatnou funkci nebo třídu, která dědí třídu *Validator*.

5.7.3 ItemAlias

Nevýhodu jednonásobné dědičnosti pomáhá zmírnit funkcionalita přidaná do PHP 5.4 s názvem *Traits*. Traity umožňují do tříd vložit definici metod ze společného zdroje, bez závislosti na dědičnosti. Hodí se to v případě, že je nutné použít stejnou metodu ve více třídách bez společného předka.

Trait s názvem *ItemAlias* byl v aplikaci použit pro vytvoření jednoduchého číselníku v rámci třídy. Ten poskytuje dvě statické metody – *itemAlias()* a *itemAliasData()*. Druhá jmenovaná slouží pro definici dat číselníku, první s tímto číselníkem pracuje.

```
use ItemAliasTrait;

public static function itemAliasData()
{
    return array(
        'stavy' => array(
            self::STAV_AKTIVNI => 'aktivní',
            self::STAV_NEAKTIVNI => 'neaktivní',
            self::STAV_REGISTRACE => 'registrace',
            self::STAV_SMAZANO => 'smazaný'
        ),
        'role' => array(
            self::ROLE_SUPERADMIN => 'Superadmin',
            self::ROLE_ADMIN => 'Administrátor',
            self::ROLE_USER => 'Uživatel'
        )
    );
}

$vsechny = Uzivatel::itemAlias('stavy');
$stav    = Uzivatel::itemAlias('stavy', $uzivatel->stav);
```

Ukázka kódu 7: Ukázka použití traitu ItemAlias, zdroj: vlastní

V příkladu je uvedena definice číselníků pro stavy a role v rámci třídy *Uzivatel*. Pro získání všech stavů z číselníku slouží první volání metody *itemAlias()* pouze s jedním parametrem. Pro získání překladu konkrétní hodnoty slouží druhé volání metody se dvěma parametry.

V aplikaci bylo této funkcionality využito především pro překlady stavů jiných hodnot, jejichž počet nebo definice se v čase nemění a není tedy nutné mít rozhraní pro jejich správu.

5.7.4 ARES

Při zakládání zákazníka je možné do fakturačních údajů uvést firmu. Pro zjednodušení tohoto kroku byla do pole *IČ* přidána možnost vyhledávání firmy v portálu ARES, který poskytuje veřejné rozhraní pro získávání detailních informací o podnikajících subjektech.

Import dat zajišťuje pomocná třída *Ares* a její metoda *nactiPodleIc()*. IČ je do třídy předáno konstruktorem a pro získání výsledku stačí zavolat výše zmíněnou metodu.

```
<?php
namespace app\components;

class Ares
{
    public $ic;

    protected $_url = 'http://wwwinfo.mfcr.cz/cgi-bin/ares/darv_rzp.cgi';

    public function __construct($ic)
    {
        $this->ic = $ic;
    }

    public function nactiPodleIc()
    {
        if (!$this->_validate()) {
            throw new \Exception("neplatna hodnota pro ico: {$this->ic}!");
        }

        $params = ['ico' => $this->ic];

        $url = $this->_sestavUrl($this->_url, $params);
        $curl = curl_init();
                curl_setopt($curl, CURLOPT_URL, $url);
                curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

        $response = curl_exec($curl);
        curl_close($curl);

        return $this->_parseResponse($response, $params);
    }

    protected function _parseResponse($response, $firma_data)
    {
        try {
            $xml = simplexml_load_string($response);

            if ($xml) {

                $ns = $xml->getDocNamespaces();
                $data = $xml->children($ns['are']);

                … dalsi zpracovani

                $firma_data['obchodni_jmeno'] = $obchodni_jmeno;
                $firma_data['mesto'] = $mesto;
                $firma_data['ulice'] = "$ulice $cislo_popisne";
                $firma_data['psc'] = $psc;
            } else {
                throw new \Exception("…");
            }
        }
        … catch blok
```

```
        // v parametru uz dostavam ico
        return $firma_data;
    }

    … dalsi metody
}
```

Ukázka kódu 8: Fragment třídy Ares, zdroj: vlastní

Ta se postará o validaci vstupu, zavolání rozhraní ARES s předaným IČ pomocí cURL a zpracování výsledku cURL dotazu, který má podobu XML dokumentu. K tomu slouží metoda *simplexml_load_string()*, která dostává v parametru string s XML, a návratem je objekt třídy *SimpleXMLElement*. Z tohoto objektu jsou načteny požadované informace a následně předány do formuláře.

Velkým problémem při získávání dat z rejstříku ARES je rychlost zpracování požadavku, která se při provozu ve špičce pohybuje v řádech desítek sekund až jednotek minut. V současné době je toto řešení dostačující, ale do budoucna je počítáno s hledáním jiného způsobu vyhledávání dat.

5.7.5 Výpočet objemu 3D modelu

Součástí práce s 3D modely je i výpočet objemu z nahrávaného souboru. Výpočet vychází z předpokladu, že platný STL model musí být tvořen trojúhelníkovou sítí, takže je možné jej rozdělit na množinu pyramid (čtyřstěnů). [28]

Při řešení výpočtu objemu a vyhledávání informací bylo nalezeno již existující řešení[15] pro programovací jazyk Python, což ve velké míře usnadnilo práci. Třída byla upravena pro jazyk PHP a na několika různých modelech ověřena její funkčnost. Ta spočívá v rozložení modelu na jednotlivé čtyřstěny, které jsou uloženy jako prvky v poli. Nad tímto polem je postupně iterováno, z každého čtyřstěnu je vypočítán objem a výsledek je postupně sčítán.

[15] http://github.com/mcanet/STL-Volume-Model-Calculator/blob/master/mesure_volume.py

```
protected function _volumeOfTriangle($v1, $v2, $v3)
{
    $v321 = $v3[1] * $v2[2] * $v1[3];
    $v231 = $v2[1] * $v3[2] * $v1[3];
    $v312 = $v3[1] * $v1[2] * $v2[3];
    $v132 = $v1[1] * $v3[2] * $v2[3];
    $v213 = $v2[1] * $v1[2] * $v3[3];
    $v123 = $v1[1] * $v2[2] * $v3[3];

    return (1.0 / 6.0) * (-$v321 + $v231 + $v312 - $v132 - $v213 + $v123);
}
```

Ukázka kódu 9: Výpočet objemu čtyřstěnu, zdroj: vlastní, [28]

Stávající funkcionalita této třídy je ale stále trochu omezená. Zatím podporuje pouze binární STL soubory (ty ale budou v systému zastoupeny téměř výhradně), ale do budoucna je počítáno i s podporou textových STL souborů. Stejně tak iterace nad jednotlivými trojúhelníky umožňuje rozšířit třídu o určení maximálních rozměrů modelu. Tato funkcionalita ovšem není pro systém důležitá, proto byla její tvorba prozatím odložena.

5.7.6 CSS styly pro mobilní menu

V kapitole 5.4.1 byl představen návrh, jak by mělo vypadat menu při zobrazení systému na malých zařízeních. Řešení se spoléhá pouze na CSS styly, pomocí JavaScriptu dochází pouze k přepnutí CSS třídy určující, zda je menu otevřené nebo ne, a k určení výšky menu. Na zdrojovém kódu níže je pro ilustraci ukázána zjednodušená struktura HTML stránky, jak je použita v systému.

```
<html>
<body>
<div class="wrap">
    <div class="wrapperLeft">
        Menu
    </div>
    <div class="wrapperMiddle">
        Obsah
    </div>
</div>
</body>
</html>
```

Ukázka kódu 10: Zjednodušená struktura HTML stránky, zdroj: vlastní

Wrap má při mobilním zobrazení 100 % šířky obrazovky, stejně tak *wrapperMiddle*. *WrapperLeft* obaluje menu a standardně není zobrazen. Cílem tedy bylo vymyslet

takovou kombinaci CSS stylů, která by zajistila zobrazení menu a odsunutí obsahu tak, aby nedošlo k jeho deformaci.

```
@media (max-width: 767px) {
    body {
        overflow-x: hidden;
        min-height: 100%;

        .wrap {
            width: 100%;
            min-height: 100%;
            overflow-x: hidden;

            .wrapperLeft {
                position: relative;
                display: block;
                float: left;
                width: 200px;
                margin-left: -200px;
                min-height: 100%;
                overflow-x: hidden;

                .transition();
            }

            .wrapperMiddle {
                float: left;
                display: block;
                width: 100%;

                .transition();
            }
        }

        &.left-open {
            overflow-x: hidden;

            .wrap {
                .wrapperLeft {
                    position: relative;
                    margin-left: 0px;
                }

                .wrapperMiddle {
                    position: relative;
                    margin-right: -100%;
                }
            }
        }
    }
}
```

Ukázka kódu 11: LESS zápis pro stylování mobilního menu, zdroj: vlastní

V ukázce zdrojového kódu můžeme vidět, že levé menu není skryto běžně používaným nastavením zobrazení na *display: none*, protože vlastnost *display* není možné použít pro

animaci pomocí CSS3 vlastnosti *transition*. Místo toho má menu nastavenou pevnou šířku a levý okraj na zápornou hodnotu šířky. Díky tomu dojde ke schování menu za levý okraj obrazovky.

Pokud uživatel klikne na tlačítko pro zobrazení menu, tagu *body* se nastaví CSS třída *left-open* a na *wrappery* se začnou aplikovat styly ze spodní třetiny LESS zápisu (Ukázka kódu 11). Menu tak dostane nulové levé odsazení, což způsobí jeho zobrazení, zatímco obsah dostane záporný pravý okraj. To zajistí prostor pro jeho posunutí vpravo a nikoliv pod menu. Vlastnost *overflow-x* nastavená na hodnotu *hidden* způsobí, že nedojde k deformaci obsahu zmenšením jeho šířky, ale k jeho skrytí za okraj obrazovky.

Využitím vlastnosti *transition* (zde schovaná v LESS mixinu *.transition()*) je celá tato operace animována.

```
.transition(@vlastnost: all, @trvani: 0.5s, @funkce: ease-out) {
    -webkit-transition: @vlastnost @trvani;
    -moz-transition: @vlastnost @trvani;
    -ms-transition: @vlastnost @trvani;
    -o-transition: @vlastnost @trvani;
    transition: @vlastnost @trvani;

    transition-timing-function: @funkce;
}
```

Ukázka kódu 12: Definice mixinu .transition(), zdroj: vlastní

6 HW a SW vybavení serveru

Součástí implementace nového informačního systému měla být i stavba a konfigurace malého serveru, na kterém by byl informační systém provozován.

Navržená konfigurace:

Základní deska	Gigabyte GA-H81N-D2H	1500 Kč
Procesor	Intel Pentium G3220 2x 3GHz	1600 Kč
Operační paměť	Crucial 2x4GB DDR3 1600 MHz	1900 Kč
SSD	Kingston HyperX 120GB	1800 Kč
HDD	Seagate Barracuda 3TB	3000 Kč
Zdroj	SeaSonic SSP-350GT	1400 Kč
Skříň	CoolerMaster Elite 120	1200 Kč
	Přibližná celková cena	**12400 Kč**

Tabulka 3: Návrh komponent pro server, zdroj cen: http://www.czc.cz

Komponenty byly voleny s ohledem na přívětivou cenu, počet současně pracujících uživatelů, spotřebu a možnou rozšiřitelnost v budoucnosti. Základní deska umožňuje osazení čtyřjádrových procesorů Intel Core i5/i7 nebo Xeon, díky přítomnosti PCIe konektoru je možné osadit například HW RAID kartu pro osazení pevnými disky do různých forem RAID pole. Zde by ovšem nastal problém při osazení procesoru Xeon, který není vybaven integrovanou grafickou kartou. To byla ovšem v případě tohoto konkrétního serveru (i s ohledem na další vývoj) velmi hypotetická situace.

Ze softwarového hlediska bylo plánované osazení virtualizačního nástroje Proxmox pro oddělení aplikačního a databázového systému. Jako operační systém bylo zvažováno mezi linuxovými distribucemi Ubuntu Server nebo Debian.

Po několika konzultacích ovšem z plánovaného záměru sešlo. Důvodem byly, kromě jednorázové finanční zátěže rozpočtu firmy, také sdílené kancelářské prostory a internetové připojení. Pro bezproblémový provoz a dostupnost informačního systému je potřebný 24h provoz serveru, veřejná IP adresa a možnost směrovat porty na routovacím zařízení, což nebylo dovoleno.

V současné době je spuštěn testovací provoz informačního systému s dokončovacími pracemi. Testovací verze je spuštěna na starším notebooku HP Compaq 6715b s následující konfigurací.

- Procesor AMD Turion X2 TL-56 1,8GHz
- Operační paměť 3GB DDR2 667 MHz
- 160GB 5400rpm pevný disk
- Integrovaná grafická karta ATi Radeon X1250

Softwarové vybavení testovacího stroje:

- Operační systém Ubuntu Server 14.04
- LAMP balíček obsahující Apache Server pro běh PHP aplikací a MySQL databázi
- PostgreSQL databáze ve verzi 9.3.6 s psql klientem také ve verzi 9.3.6
- OpenSSH server pro vzdálený přístup
- Verzovací systém Git
- Doplňky PHP, například rozšíření Pecl, mcrypt nebo cURL

Z prozatímního testovacího provozu bylo zjištěno, že tato konfigurace pro provoz systému plně dostačuje, jedinou náročnější součástí systému je nahrávání, resp. konkrétně parsování 3D modelu při jeho ukládání. Ovšem tento proces proběhne u každého modelu vždy pouze jednou, zobrazování modelu už obsluhuje prohlížeč na klientské straně.

Pro produkční provoz bylo nakonec vybráno řešení v podobě pronájmu VPS, tedy virtuálního serveru. Zvolena bude obdobná (ale výkonnější) konfigurace, jako je na testovacím stroji, stejně jako softwarové vybavení doplněné o antivirové řešení a firewall. Výhodou tohoto řešení je garantovaná HW dostupnost služby provozovaná profesionální firmou, nízká zátěž pro firemní rozpočet a nutnost starat se o server pouze po stránce softwaru. Po spuštění informačního systému bude na virtuální server migrován i firemní web.

Předpokládaná HW výbava virtuálního stroje:

- Dvoujádrový procesor, alespoň 2,5 GHz takt na jádro
- 3 až 4 GB RAM
- 100 GB pevný disk

Po spuštění informačního systému do produkčního provozu bude nutné také nastavit pravidelné zálohování dat. To se týká především databáze, která by se měla zálohovat

ideálně vícekrát denně, ovšem v tomto případě bude dostačovat provést zálohu každý den v nočních hodinách, kdy se neočekává žádný provoz v systému. Denní zálohy budou udržovány po dobu dvou měsíců. Každý poslední den v měsíci budou denní zálohy předchozího měsíce promazány a zůstanou například jen z posledního dne daného měsíce.

Pokud budeme uvažovat náklady za pronájem VPS v hodnotě 500 Kč / měsíc, vlastní řešení serveru se začíná vyplácet po 25 měsících. Do těchto nákladů ovšem není započítána spotřeba energie, náklady na obnovu HW při jeho selhání nebo pokrytí výpadků serveru. V tomto ohledu začíná vycházet VPS řešení lépe. Výhodou vlastního řešení je větší prostor na pevném disku, více operační paměti, ale především virtualizace systémů – oddělení aplikační a databázové vrstvy.

7 Shrnutí výsledků

V první části diplomové práce byly popsány teoretické základy pro vývoj informačního systému. Na tyto základy navazuje praktická část, která se věnuje analýze firmy a požadavků na systém. Z této analýzy vznikly nejprve návrhy a poté se postupně přešlo na samotnou implementaci informačního systému. Protože se jedná o mladou a začínající firmu, nebyly požadavky na systém náročné. Do budoucna se však předpokládá, že tyto požadavky se budou s růstem firmy zvyšovat.

Nově vytvářený informační systém by měl především odpovídat požadavkům zadavatele, proto během vývoje probíhala úzká spolupráce se zástupci firmy. Velký důraz byl také kladen na jednoduchost používání informačního systému nejen na počítačích, ale i tabletech a jiných mobilních zařízeních, které jsou ve firmě využívány.

Součástí práce měl být také návrh, stavba a konfigurace serveru, který měl sloužit pro běh vytvářeného systému. K tomuto kroku nakonec nedošlo a bylo vybráno náhradní řešení v podobě pronájmu virtuálního serveru.

V současné době není systém plně dokončen a probíhají dokončovací práce na některých modulech systému. Zbývá dokončit moduly Fakturace a Kalendář a napojit systém na mailovou službu, která by rozesílala informace zákazníkům a notifikace uživatelům systému. Důvodem absence zmíněných částí je neodhadnutí celkové náročnosti systému a také vývoj probíhající pouze v jednom člověku.

Aktuálně je na adrese http://test.senovo.cz/inter_is/web vystavena beta verze pro testovací provoz a připomínkování dokončených částí.

7.1 Možnosti budoucího vývoje

Vytvořený informační systém autor plánuje i nadále spravovat a rozvíjet. Již v průběhu implementací vznikly podněty na novou funkcionalitu nebo rozšíření stávajících. Mimo jiné se jedná o exporty dat pro účetní systém. Nové funkce ovšem budou přidány až po dokončení systému nad rámec této práce.

Do budoucna je také plánované spuštění vlastního eshopu, který by byl s informačním systémem propojen.

8 Závěry a doporučení

Diplomová práce se zabývá návrhem a implementací responzivního informačního systému pro malou firmu. Tento návrh vychází z analýzy firmy, současného stavu a požadavků zástupců firmy.

Hlavními cíli práce byly návrh a implementace informačního systému. Tyto cíle byly naplněny a vznikl tak funkční informační systém, který by měl firmě přinést lepší organizaci a efektivitu práce. Zástupci firmy oceňují jednoduché a přívětivé uživatelské prostředí, se kterým se bude uživatelům snadno pracovat. Jednodušší a efektivnější práce s informačním systémem by firmě měla umožnit soustředit se na ostatní aspekty podnikání, jako například získávání nových zákazníků nebo rozšiřování firemního portfolia.

Informační systém bude ve firmě sloužit pro správu vytvořených 3D modelů, zadávání objednávek, tvorbu faktur, správu zákazníků a jednoduchou agendu v podobě kalendáře s úkoly a schůzkami.

Součástí práce měla být i stavba a konfigurace serveru, na kterém by navržený systém běžel. Tohoto cíle bylo dosaženo pouze z části. Byl vytvořen návrh komponent pro stavbu serveru, ale k vlastní stavbě a konfiguraci z finančních a provozních důvodů nedošlo.

V průběhu analýzy bylo řešeno několik drobných problémů. Mezi tyto problémy je možné zařadit nejasné představy o funkcionalitě systému nebo neexistující firemní procesy. V průběhu vlastní implementace nebyly řešeny žádné zásadní problémy. Naopak, jednotná softwarová platforma (především jádro Blink v rámci webových prohlížečů) velmi zjednodušila proces optimalizace vzhledu systému.

Protože je implementace systému součástí této práce, vznikají pro firmu náklady pouze v podobě paušálu za pronájem VPS, které jsou odhadovány na cca 500 Kč / měsíc. Suma těchto nákladů sice může v horizontu několika let převýšit prvotní investici do vlastního serveru, ale pro firmu je toto řešení ve stávající chvíli vhodnější.

Po celkovém dokončení informačního systému jej autor plánuje i nadále rozvíjet a spravovat.

9 Seznam použitých zdrojů

1. **Klimeš, Cyril.** *Informační systémy, texty pro distanční studium.* Nitra : Univerzita Konštantína Filozofa, 2006.

2. **Kučerová, Helena.** *Projektování informačních systémů.* Praha : Vyšší odborná škola informačních služeb, 2007.

3. **Čech, Pavel a Bureš, Vladimír.** *Podniková Informatika.* 1. vydání. Hradec Králové : GAUDEAMUS, 2009.

4. **Sodomka, Petr a Klčová, Hana.** *Informační systémy v podnikové praxi.* 2. aktualizované a rozšířené vydání. Brno : Computer Press, a.s., 2010. ISBN 978-80-251-2878-7.

5. SWOT analýza. *Středoevropské centrum pro finance a management.* [Online] Středoevropské centrum pro finance a management, 2005 - 2012. [Citace: 14. leden 2015.] http://www.finance-management.cz/080vypisPojmu.php?IdPojPass=59&X=SWOT+analyza/.

6. **Zikmund, Martin.** Kde se vzala a k čemu je PEST analýza. *BusinessVize.* [Online] 29. listopad 2010. [Citace: 25. březen 2015.] http://www.businessvize.cz/planovani/kde-se-vzala-a-k-cemu-je-pest-analyza.

7. **Koch, Miloš.** Posouzení efektivnosti informačního systému. *Trendy ekonomiky a managementu.* Brno : Vysoké učení technické v Brně, 2013.

8. **Kabálek, Vitězslav.** Nástroje byznys modelování. *Bakalářská práce.* Brno : Masarykova univerzita, 2009.

9. **Šmíd, Vadimír.** Životní cyklus informačního systému. [Online] [Citace: 8. únor 2015.] http://www.fi.muni.cz/~smid/mis-zivcyk.htm.

10. **Kajzar, Dušan.** *Tvorba informačních systémů II.* Opava : Slezská univerzita v Opavě, 2005. ISBN 80-7248-288-2.

11. **Molnár, Zdeněk.** *Moderní metody řízení informačních systémů.* Praha : Grada, 1992. ISBN 80-85623-07-2.

12. **Kerner, Aleš.** Co byste měli vědět o responzivním designu. *Interval.* [Online] 27. červen 2013. [Citace: 7. únor 2015.] http://interval.cz/clanky/co-byste-meli-vedet-o-responzivnim-designu/.

13. **Škrášek, Jan.** Nástroje responzivního webdesignu. *Programujte.com.* [Online] 12. srpen 2013. [Citace: 7. únor 2015.] http://programujte.com/clanek/2013062900-nastroje-responzivniho-webdesignu/.

14. **Michálek, Martin.** CSS pixel. *Vzhůru dolů.* [Online] 10. září 2014. [Citace: 8. únor 2015.] http://www.vzhurudolu.cz/prirucka/css-pixel.

15. CSS Pixel Widths. *CANbike.* [Online] 7. říjen 2014. [Citace: 8. únor 2015.] http://www.canbike.org/CSSpixels/.

16. Katalog mobilů. *MobilMania.* [Online] Mladá Fronta. [Citace: 8. únor 2015.] http://www.mobilmania.cz/katalog-mobilu/sc-63-c-1/default.aspx.

17. **Michálek, Martin.** (Snad už) Definitivní responzivní obrázky – srcset a sizes. *Zdroják.* [Online] 16. leden 2015. [Citace: 10. únor 2015.] http://www.zdrojak.cz/clanky/snad-uz-definitivni-responzivni-obrazky-srcset-sizes/.

18. **Štráfelda, Jan.** HTML. *Adaptic.* [Online] Adaptic, s.r.o. [Citace: 30. březen 2015.] http://www.adaptic.cz/znalosti/slovnicek/html/.

19. —. Kaskádové styly. *Adaptic.* [Online] Adaptic, s.r.o. [Citace: 30. březen 2015.] http://www.adaptic.cz/znalosti/slovnicek/kaskadove-styly/.

20. **Yii Software LLC.** The Definitive Guide to Yii 2.0. *Yii Framework.* [Online] 2014. [Citace: 22. březen 2015.] http://www.yiiframework.com/doc-2.0/guide-index.html.

21. Yii Framework 2.0 API Documentation. *Yii Framework.* [Online] Yii Software LLC, 21. březen 2015. [Citace: 15. duben 2015.] http://www.yiiframework.com/doc-2.0/index.html.

22. **Safronov, Mark a Winesett, Jeffrey.** *Web Application Development with Yii 2 and PHP.* Birmingham : Packt Publishing Ltd., 2014. ISBN 978-1-78398-188-5.

23. **Venners, Bill.** Orthogonality and the DRY Principle. *Artima Developer.* [Online] 10. březen 2013. [Citace: 22. březen 2015.] http://www.artima.com/intv/dry.html.

24. **Tutorial Republic.** Bootstrap Introduction. *Tutorial Republic.* [Online] 2015. [Citace: 22. březen 2015.] http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-introduction.php.

25. **Stěhule, Pavel.** PostgreSQL. *PostgreSQL.* [Online] 2015. [Citace: 12. březen 2015.] http://postgres.cz/wiki/PostgreSQL.

26. **Vrána, Jakub.** Srovnání funkcí MySQL a PostgreSQL. *PHP triky.* [Online] 17. červen 2009. [Citace: 12. březen 2015.] http://php.vrana.cz/srovnani-funkci-mysql-a-postgresql.php.

27. **Malý, Martin.** LESS: stejné CSS za méně peněz. *Zdroják.* [Online] 21. říjen 2010. [Citace: 22. březen 2015.] http://www.zdrojak.cz/clanky/less-stejne-css-za-mene-penez/.

28. How do I calculate the volume of an object stored in STL files? *StackOverflow.* [Online] 1. červenec 2011. [Citace: 21. listopad 2014.] http://stackoverflow.com/questions/6518404/how-do-i-calculate-the-volume-of-an-object-stored-in-stl-files.

29. **Vondrák, Ivo.** *Metody byznys modelování.* Ostrava : VŠB – Technická univerzita Ostrava, 2004.

10 Seznam obrázků

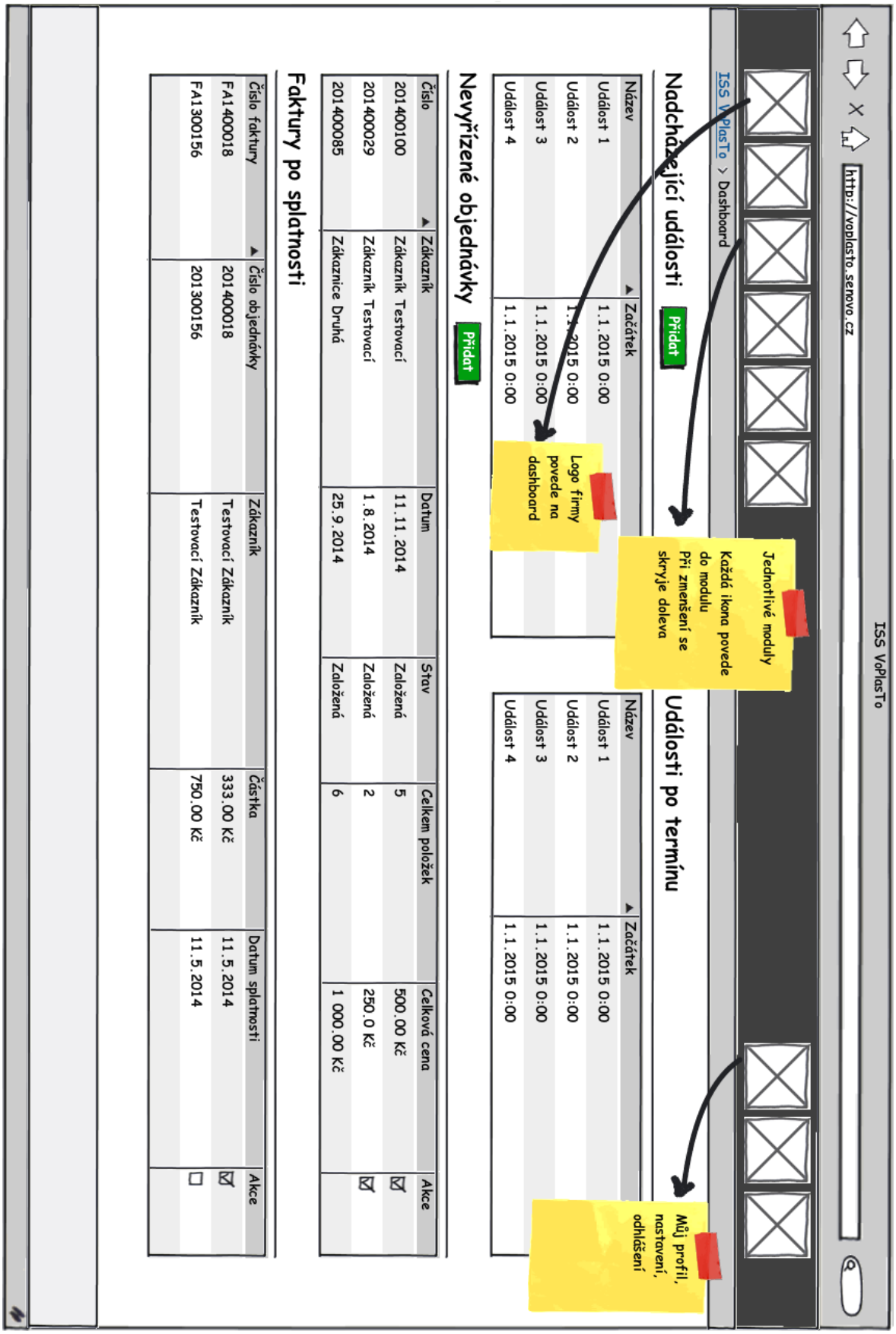
11 Seznam tabulek

12 Seznam ukázek zdrojových kódů

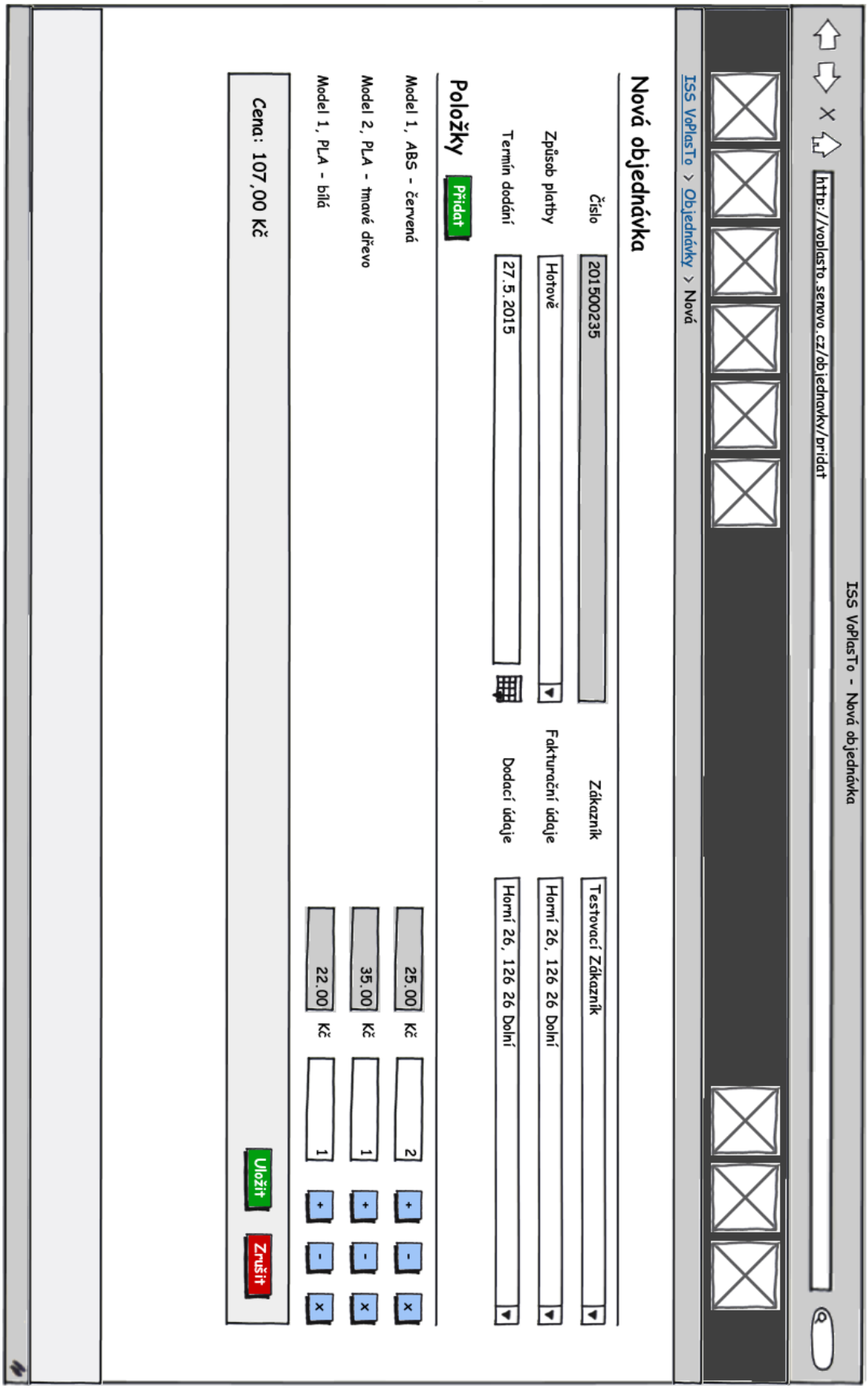
13 Přílohy

Příloha 1: EPC diagram životního procesu objednávky, zdroj: vlastní

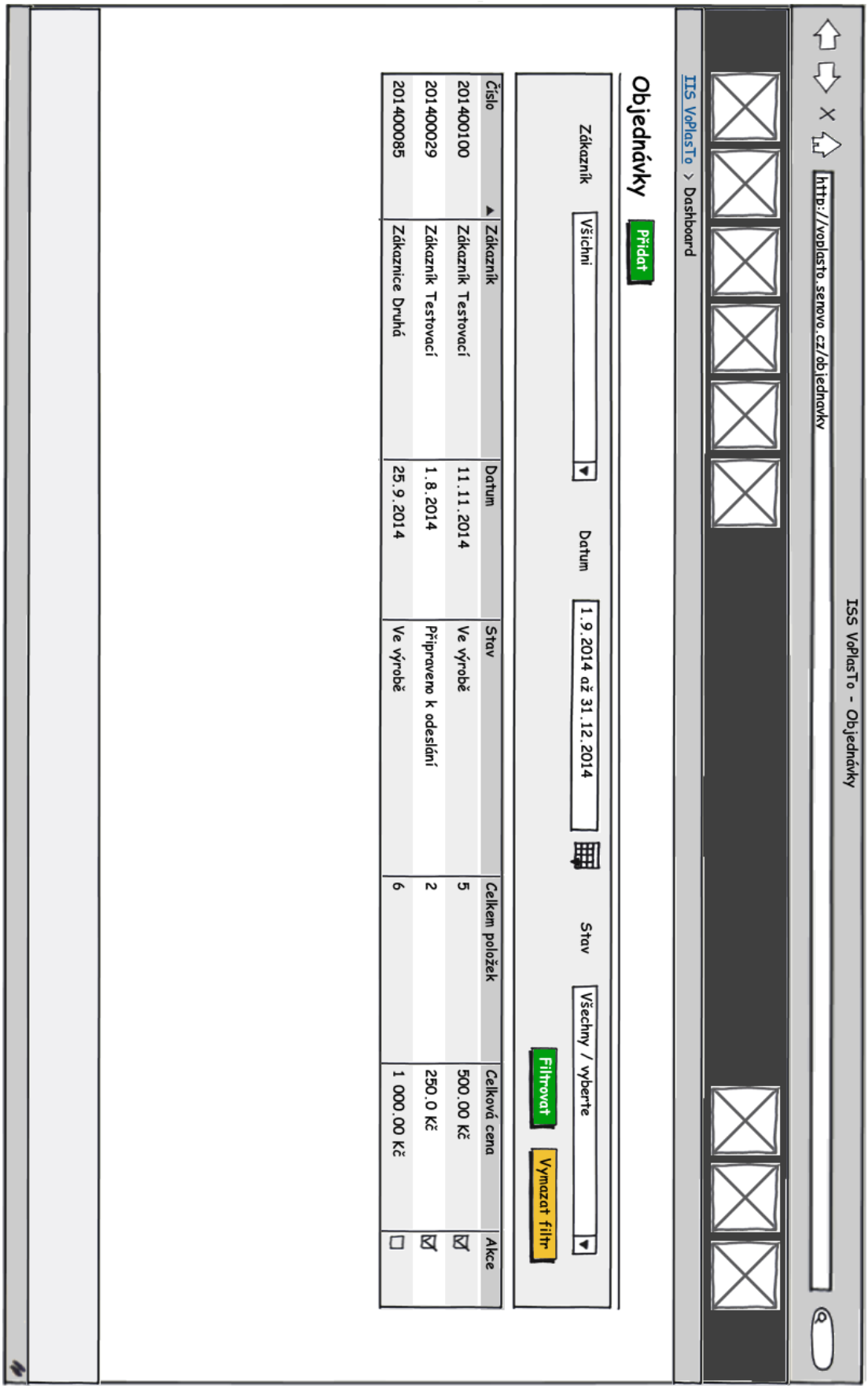
Příloha 2: Wireframe pro hlavní obrazovku, zdroj: vlastní



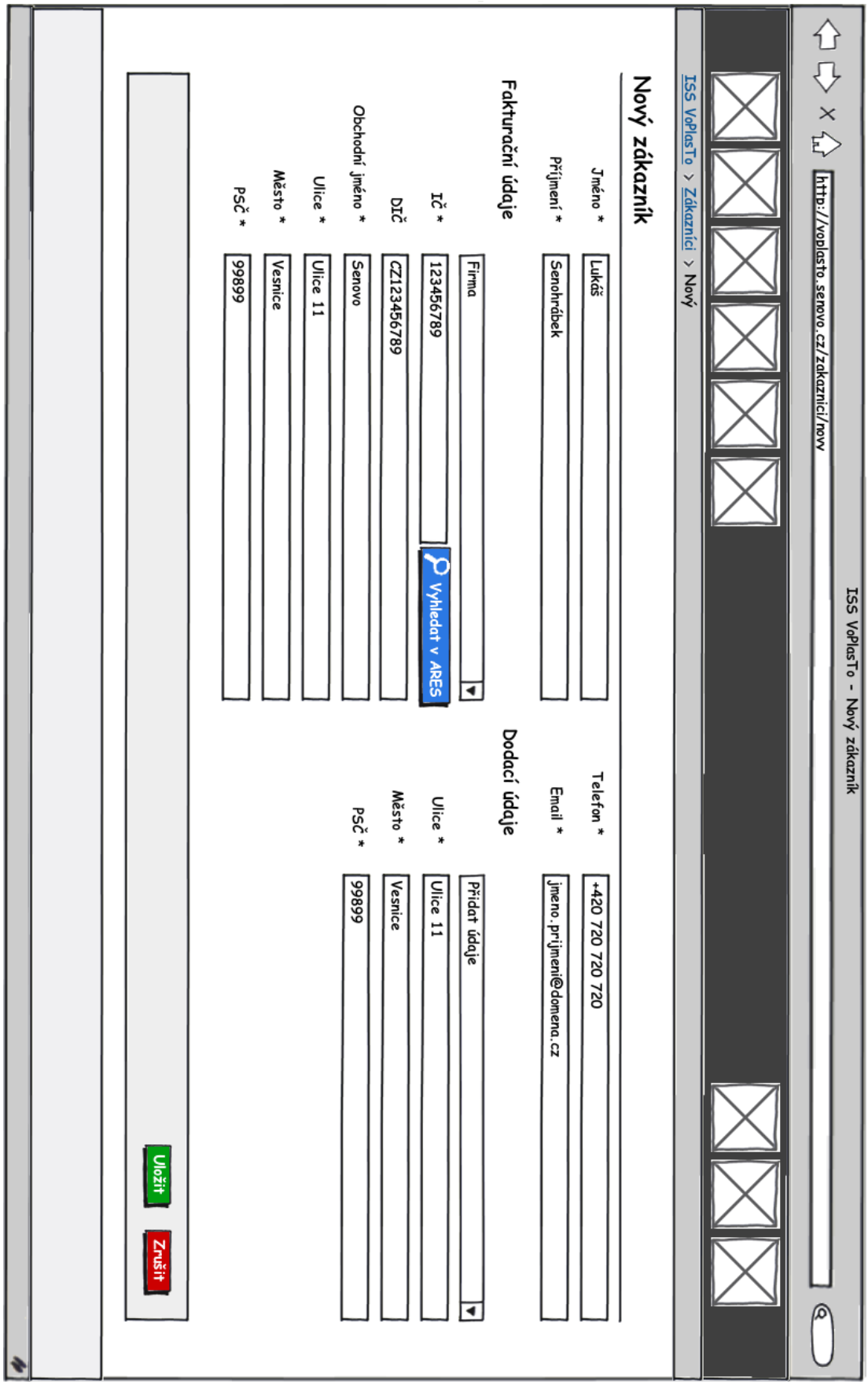
Příloha 3: Wireframe formuláře pro novou objednávku, zdroj: vlastní



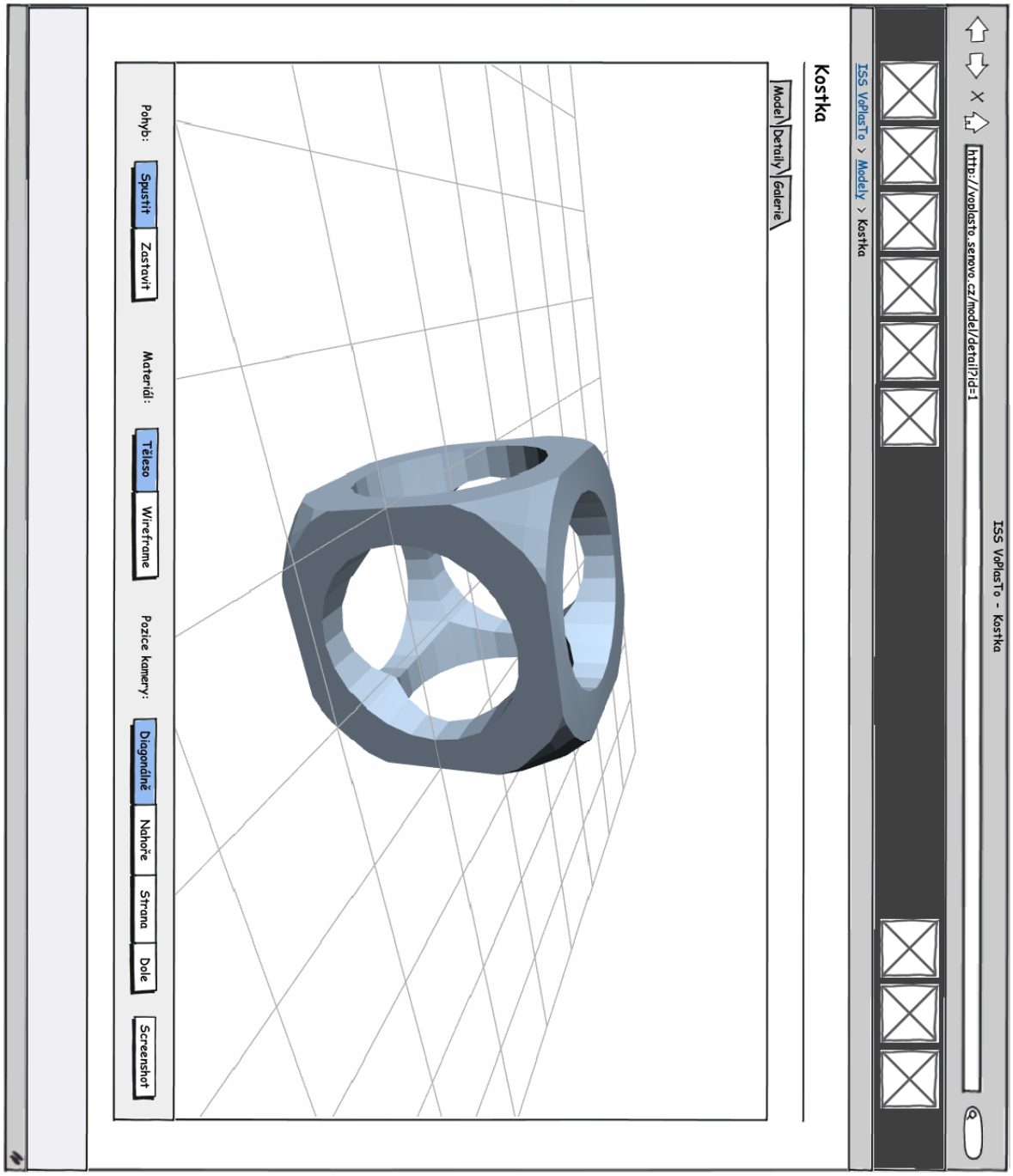
Příloha 4: Wireframe seznamu objednávek, zdroj: vlastní



Příloha 5: Wireframe formuláře pro nového zákazníka, zdroj: vlastní



Příloha 6: Wireframe detailu 3D modelu, zdroj: vlastní



Příloha 7: Wireframe hlavní obrazovky na mobilním zařízení, zdroj: vlastní

Příloha 8: Implementace třídy pro práci se STL soubory

```
<?php
class STLUtils
{
    protected $_file;
    protected $_density;
    protected $_triangles = array();
    protected $_volume = null;
    protected static $_instance;

    public function __construct($file, $density = 1.04)
    {
        if (!file_exists($file)) {
            throw new Exception("File not exists: $file!");
        } elseif (!is_readable($file)) {
            throw new Exception("File is not readable: $file!");
        } else {
            $this->_file = $file;
            $this->_density = $density;

            $this->_load();
        }
    }

    public static function getInstance($file, $density = 1.04)
    {
        if (null == self::$_instance) {
            self::$_instance = new self($file, $density);
        }

        return self::$_instance;
    }

    protected function _load()
    {
        if ($this->_file == null) {
            throw new Exception("No file to load!");
        }

        try {
            $fp = fopen($this->_file, "rb");
            $header = file_get_contents($this->_file, null, null, 0, 79);

            fseek($fp, 80);
            $data = fread($fp, 4);
            $total = unpack("I", $data)[1];

            for ($i = 0; $i < $total; $i++) {
                $this->_triangles[] = array(
                    'N'  => unpack("f3", fread($fp, 12)),
                    'V1' => unpack("f3", fread($fp, 12)),
                    'V2' => unpack("f3", fread($fp, 12)),
                    'V3' => unpack("f3", fread($fp, 12)),
                    'B'  => unpack("S",  fread($fp, 2))[1]
                );
            }
        } catch (\Exception $e) {
            throw new STLException($e->getMessage());
```

```
        }
    }

    protected function _calculateVolume()
    {
        $volume = null;
        foreach ($this->_triangles as $triangle) {
            $volume += $this->_volumeOfTriangle($triangle['V1'], $triangle['V2'],
$triangle['V3']);
        }

        return $volume;
    }

    protected function _volumeOfTriangle($v1, $v2, $v3)
    {
        $v321 = $v3[1] * $v2[2] * $v1[3];
        $v231 = $v2[1] * $v3[2] * $v1[3];
        $v312 = $v3[1] * $v1[2] * $v2[3];
        $v132 = $v1[1] * $v3[2] * $v2[3];
        $v213 = $v2[1] * $v1[2] * $v3[3];
        $v123 = $v1[1] * $v2[2] * $v3[3];

        return (1.0 / 6.0) * (-$v321 + $v231 + $v312 - $v132 - $v213 + $v123);
    }

    public function getVolume()
    {
        if ($this->_volume == null) {
            $this->_volume = $this->_calculateVolume() / 1000;
        }

        return $this->_volume;
    }

    public function getWeight()
    {
        return $this->_volume * $this->_density;
    }

    public function __get($name)
    {
        switch ($name) {
            case 'volume':
                return $this->getVolume();
                break;
            case 'weight':
                return $this->getWeight();
                break;
            default:
                throw new Exception("Unknown property: $name");
        }
    }
}
```

Příloha 9: Implementace dynamického ceníku pro využití v komponentě DetailView z Yii frameworku

```
<?php
class CenikModelu extends Model
{
    private $_data = [];
    public $model_pk;

    public function __construct($config = [])
    {
        parent::__construct($config);

        if (isset($config['model_pk'])) {
            $sql = "
              SELECT
                  replace(lower(unaccent(mv.nazev || '_' || mv.barva)), ' ', '_')
AS id
                , mv.nazev || ' - ' || mv.barva AS \"label\"
                , round(coalesce((SELECT objem FROM model3d WHERE model_pk = :mpk)
* mv.hustota * mv.cena, 0.00), 2) AS cena
              FROM material_view mv
            ";

            $data = \Yii::$app->db->createCommand($sql)->bindValue('mpk',
$config['model_pk'])->queryAll();

            foreach ($data as $radek) {
                $this->_data[$radek['id']] = ['label' => $radek['label'], 'cena'
=> $radek['cena']];
            }
        }
    }

    public function __get($attr)
    {
        if (in_array($attr, $this->attributes())) {
            return HtmlPurifier::process(
                number_format(
                    $this->_data[$attr]['cena'],
                    2,
                    ',',
                    ' '
                ) . ' ' . \Yii::$app->formatter->currencyCode
            );
        } else {
            return parent::__get($attr);
        }
    }

    public function attributes()
    {
        return array_keys($this->_data);
    }

    public function getAttributes()
    {
        $attrs = array();

        foreach ($this->attributes() as $attr) {
```

```
            $attrs[$attr] = $this->_data[$attr]['cena'];
        }

        return $attrs;
    }

    public function attributeLabels()
    {
        $labels = array();

        foreach ($this->attributes() as $attr) {
            $labels[$attr] = $this->_data[$attr]['label'];
        }

        return $labels;
    }

    public function vratData()
    {
        return $this->_data;
    }
}
```

Příloha 10: Zadání závěrečné práce

FIM UHK
UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **Lukáš Senohrábek**

Obor studia: Aplikovaná informatika (2)
Jméno a příjmení vedoucího práce: **Filip Malý**

Název práce:
Responzivní informační systém pro malou firmu

Název práce v AJ:
Responsive information system for small business

Podtitul práce:

Podtitul práce v AJ:

Cíl práce: Návrh a implementace nového responzivního informačního systému pro malou firmu.

Osnova práce:
1. Úvod
2. Analýza firmy a požadavků
3. Technologie využité ve vlastní práci
4. HW a SW výbava serveru
5. Návrh a implementace IS
6. Závěr

Projednáno dne: 23.10.2014

Podpis studenta

Podpis vedoucího práce