Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Webová grafická aplikace pro řešení 2D řezného problému

Petr Kůla

Bakalářská práce
2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2018/2019**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Kůla**
Osobní číslo:	**I16110**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Webová grafická aplikace pro řešení 2D řezného problému**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je navrhnout a implementovat webovou aplikaci pro řešení řezného problému z oblasti lineárního programování. Konkrétně jde o řešení 2D řezného problému - aplikace umožní zadání geometrického tvaru s rozměry pro výplň a také umožní zadat prvky, kterými bude plocha zaplněna. Následně dojde k výpočtu s cílem minimalizovat odpad při výplni definované plochy.

Rozsah grafických prací:

Rozsah pracovní zprávy: **30-40 normostran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

VOLEK, Josef. Operační výzkum I. Vyd. 2., nezměn. Pardubice: Univerzita Pardubice, 2008. ISBN 978-80-7395-073-6
CHINNATHAMBI, Kirupa. Learning React 1st Edition. Addison-Wesley Professional, 2016. ISBN 978-0134546315

Vedoucí bakalářské práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2018**

Termín odevzdání bakalářské práce: **12. května 2019**

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 20. března 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 9. 5. 2019

Petr Kůla

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu práce doc. Ing. Michaelu Bažantovi, Ph.D. za cenné rady a vedení při zpracování této práce. Dále bych chtěl poděkovat všem autorům, jejichž díla jsem při zpracování této práce použil. A v neposlední řadě bych chtěl poděkovat rodině za podporu během celého studia.

ANOTACE

Bakalářská práce je věnována problematice řezného problému. Byla navržena webová aplikace s grafickým a textovým výstupem umožňující optimálně umístit tvary s cílem minimalizovat odpad materiálu. V teoretické části práci se nachází popis a rozdělení řezného problému, v praktické části je popis použitých technologií a implementace webové aplikace.

KLÍČOVÁ SLOVA

řezný problém, webová aplikace, strip packing, bin packing, PHP, HTML

TITLE

Web graphic application for 2D cutting stock problem solution

ANNOTATION

The bachelor thesis is devoted to the cutting stock problem. A web-based application with graphical and text output has been designed to optimally position shapes to minimize material waste. In the theoretical part of the thesis, there is description and division of cutting stock problem, in practical part, there is description of used technologies and implementation of web application.

KEYWORDS

cutting stock problem, web application, strip packing, bin packing, PHP, HTML

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

1D	jednodimenzionální
2D	dvoudimenzionální
3D	třídimenzionální
CSS	Cascading Style Sheets
DH	klesající výška
DW	klesající šířka
DOM	Document Object Model
GUI	grafické uživatelské rozhraní
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
PHP	Hypertext Preprocessor
SGML	Standard Generalized Markup Language
WWW	World Wide Web

ÚVOD

Bakalářská práce se zabývá tvorbou webové aplikace pro řešení řezného problému, přesněji jeho dvojrozměrné varianty. V české literatuře se jedná o velice špatně popsané téma. Veškeré informace končí u jednorozměrné varianty řezného problému, i když se jedná o velice zajímavý a užitečný problém, který pomáhá k úspoře materiálu a tím ke snížení nákladů na výrobu. Práce je rozdělena na dvě části – teoretickou a praktickou.

První část práce obsahuje popis řezného problému. Je zde uvedeno, co je řezný problém, jeho cíl a základní rozdělení podle různých aspektů a přístup k výpočtu, kde jsou popsány silné a slabé stránky každé varianty. Teoretická část dále obsahuje popis tří základních algoritmů pro heuristický přístup k výpočtu, které jsou popsány jak slovně, tak pomocí vývojových diagramů. Na konci této části jsou uvedeny již existující programy pro výpočet řezného problému. Jsou porovnány jejich silné a slabé stránky a jejich základní popis.

Ve druhé části je popsána implementace řezného problému do webové aplikace. Nejprve jsou popsány technologie, se kterými jsem pracoval s uvedením jejich stručné historie. Dále se nachází popis samotné webové aplikace, její adresářová struktura a popis grafického uživatelského rozhraní. Na závěr kapitoly je uvedeno porovnání, ve kterém jsem porovnával správnost nářezového plánu s nářezovým plánem od jiného autora založeném na stejné sadě dat a porovnání nářezového plánu, který vznikl heuristickou metodou s nářezovým plánem, který vznikl přesnou metodou.

1 TEORETICKÁ ČÁST

Výsledkem řezného problému je vždy nářezový plán. Ten je zadán dvěma skupinami objektů, v první skupině se nachází materiál, který bude použit pro výrobu, zpravidla se jedná o obdélníkové desky. Ve druhé skupině objektů se nachází rozměry výrobků, které se budou z výchozího materiálu vyrábět. Cílem je najít co nejlepší řešení, za to je považováno to, při kterém vznikne co nejmenší odpad. Ten je definován jako část materiálu, ze kterého jsou vyřezávány výrobky a nelze ho použít pro další výrobu – další výrobky již nejdou z materiálu vyřezat a materiál tedy nemá další uplatnění. Řezný problém se hojně využívá v průmyslu při výrobě, kde pomáhá podnikatelům šetřit jejich náklady díky úspoře materiálu. Používá se například ve sklářství, výrobě plošných spojů, nebo textilním průmyslu.

1.1 Rozdělení řezného problému

Řezný problém lze rozdělit na několik kategorií podle jeho vlastností a přístupu. Je například zbytečné počítat vícedimenzionální řezný problém, než v jaké dimenzi potřebujeme výsledný nářezový plán. Dále je většinou zbytečné počítat nářezový plán pomocí exaktních algoritmů, protože výsledný nářezový plán se v množství odpadu příliš často neliší, vývoj programu a samotný výpočet zabere více času a peněz, než kolik se ušetří pomocí heuristického řešení.

1.2 Rozdělení podle dimenze

První rozdělení řezného problému je podle dimenze, ve které řezný problém figuruje. Zpravidla se jedná o tři základní dimenze Euklidova prostoru – 1D, 2D a 3D. Soustava může být i vícerozměrná – 4D, nebo 5D. Jedná se o 3D soustavu doplněnou o další rozměr. Další rozměr může být čas, nebo určitá priorita výroby. S vícerozměrnými soustavami se lze setkat jen velmi zřídka.

1.2.1 Jednorozměrný řezný problém

Nejjednodušší varianta řezného problému, kde se počítá pouze s jedním rozměrem materiálu – délkou, ostatní rozměry jsou zanedbány. Tato varianta se používá zejména při dělení drátu, nebo dřevěných latí, kde je šířka a výška jednotlivých dílců stejná. I přes všechno se může zdát, že výpočet řezného plánu je jednoduchý, není to pravda. Výpočetní složitost roste exponenciálně s počtem výsledných tvarů, protože pro každý dílec je počítána kombinace se všemi ostatními dílci určenými pro výrobu. Optimalizace, kterou lze docílit menšího počtu

kombinací se nazývá *Problém batohu.* Tento problém se poté řeší pomocí metody *Branch and bound*. [3]

Tato varianta řezného problému má uplatnění především při vyrábění drátů pro elektrickou síť, kde je cílem, aby drát byl co nejméně napojován a v neposlední řadě také úspora materiálu. [2]

1.2.2 Dvourozměrný řezný problém

Tento řezný problém je oproti 1D problému rozšířen o šířku materiálu. Jedná se o nejčastěji používanou variantu problému. Důvodem je, že pro většinu řešených problému jsou dvě dimenze plně dostačující a třetí dimenze není použita, nebo ji lze zanedbat. Dalším důvodem je, že bez řezného plánu je poměrně složité najít nejlepší řešení, ale zároveň výpočtový čas a programování není tak složité, jako u 3D řezného problému. Využití se nachází při výrobě skla, nábytku, řezání desek plošných spojů, papíru, plechu atd. [2]

1.2.3 Třírozměrný řezný problém

Třírozměrný řezný problém reprezentuje tvary z reálného světa. Tvar je tedy zadán všemi třemi rozměry. Výpočet i programování jsou náročné a nelze jednoduše odvodit, existuje mnoho výzkumů, které se zabývali aproximací při výpočtu, ale algoritmy jsou stále výpočetně složité. O tomto problému nelze hovořit jako o řezném problému, protože řezání třírozměrných objektů z výchozího objektu je velice náročné. Proto se tento problém označuje pojmem *„Bin packing problem“*. [2]

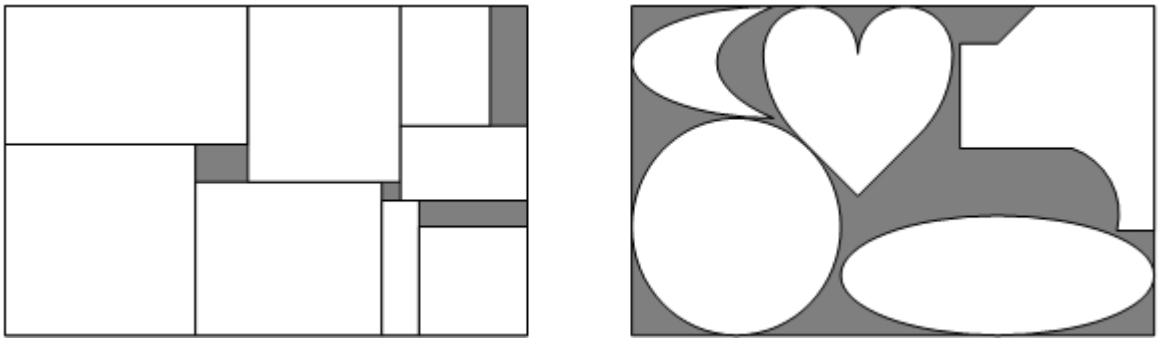
Z názvu vyplývá, že se nejedná o řezný problém, ale o problém, který se týká kontejnerů. Použití tohoto problému je zejména u dopravních společností, které tímto způsobem řeší maximální využití místa při přepravě v kontejnerech, nebo kamionech.

1.3 Rozdělení podle tvaru

V literatuře se lze setkat s algoritmy rozdělenými podle tvaru objektů a základny, na kterou jsou pomocí výpočtů umisťovány. Základní rozdělení je tedy dvojí. V prvním případě se jedná regulární tvary, v literatuře označováno jako *„regular shapes“*. Tímto pojmem jsou označovány tvary obdélníkového rázu. Jde tedy o čtverce a obdélníky. Ve většině případů tento přístup stačí, protože je výpočtově i implementačně nejjednodušší a většina tvarů, které chceme umístit na základnu je obdélníkového tvaru.

Pokud obdélníkové tvary nestačí, lze se setkat ještě s neregulárními tvary. V literatuře jsou označovány jako *„irregular shapes“*, v češtině se používá výraz neregulární, nebo neobecné

tvary. Jedná se o všechny ostatní tvary, které nejsou obdélníkové, to je například kruh, hvězda, nebo polygon.



Obrázek 1- Regulérní a neregulérní tvary

V praxi se výsledek řezného problému nepočítá z rovnic neregulárních tvarů, kterými jsou zadány, ale umístí se do obdélníku, který nejlépe odpovídá tvaru předmětu, nebo se ohraničí polygonem. Díky tomu se zjednoduší výpočet. Dále může mít i základna, na kterou se tvary umisťují neregulérní tvar, to se používá zejména v textilním průmyslu, kde by jinak vznikal velký odpad. [2]

1.4 Rozdělení podle omezení materiálu

1.4.1 Strip packing problem

Omezení materiálu je dvojí. První omezení omezuje šířku materiálu, ale nikoliv výšku. Výška je buď nekonečná, nebo dostatečně dlouhá, takže ji můžeme zanedbat. Materiál je nejčastěji role pravidelného tvaru, nebo pás. Z toho vyplývá označení v anglicky psané literatuře pro tento problém – *Strip packing problem*. Cílem tohoto problému je minimalizovat výšku použitého materiálu. Uplatnění se nachází zejména v průmyslu, kde pracují s plechem, nebo v textilním průmyslu, kdy v obou těchto případech je materiál dodáván ve formě role.

Tvary, které můžou být pravidelných, či nepravidelných rozměrů jsou umisťovány do řádků a nesmí se navzájem překrývat. Algoritmy pro strip packing nejčastěji pracují s řádky, tento druh algoritmů se v angličtině označuje termínem level-oriented.

U různých variant těchto algoritmů se používají zkratky DH, nebo DW, to jsou zkratky pro anglické termíny decreasing height a decreasing width. To znamená, že tvary jsou na začátku seřazeny sestupně podle výšky, nebo podle šířky. Nejčastěji se používá řazení podle výšky.

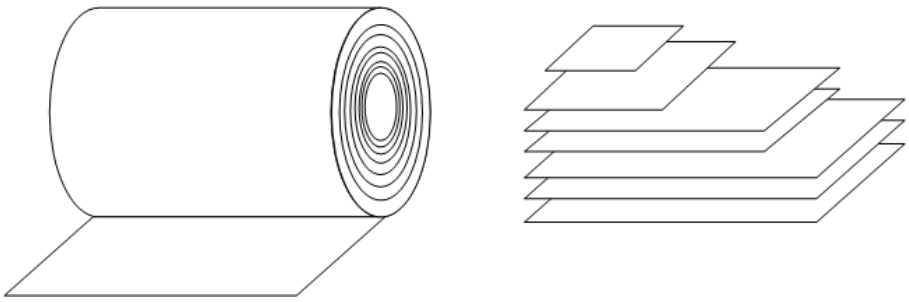
Tento problém lze nadále rozdělit na statický a dynamický. Při statické variantě jsou na začátku zadány tvary, které vyřežou a v průběhu provádění nejsou přidávány další tvary.

Naopak v dynamické variantě tohoto problému jsou při provádění přidávány další nové tvary. Při přidání nového tvaru dojde k přepočtu ještě nevyřezaných tvarů. Dynamická varianta je tedy rychlejší a okamžitě reaguje na změny. Neposkytuje ale tak přesné výsledky jako statická varianta. [2]

1.4.2 Bin packing problem

Toto je druhé omezení materiálu. Na rozdíl od *strip packing problem* je brána v potaz i výška materiálu. Materiál je dostupný nejčastěji v dílcích ve tvaru obdélníku. Tyto dílce nemusí být vždy stejných rozměrů. Úkolem tohoto problému je vybrat takové dílce výchozího materiálu, aby odpad byl co nejmenší. Název tohoto problému vznikl na základě dílců výchozího materiálu, které jsou nazývány kontejnery, proto *bin packing problem*. [2]

Mnohdy je lepší a výhodnější použít tuto variantu problému než *strip packing problem*. Důvodem je výchozí materiál, protože často by byla celá role zbytečná, proto se vyplatí nakoupit od dodavatele již nařezaný materiál i za cenu většího odpadu. Dále je tento problém lepší, pokud často řežeme několik stejných tvarů – set tvarů. V tomto případě se vyplatí vytvořit řezný plán pro tento set a nechat si od dodavatele materiálu dodávat obdélníky pro daný set. Protože často dodavatelé poskytují množstevní slevu při větším odběru materiálu. [2]



Obrázek 2- Materiál pro strip packing a pro bin packing

1.5 Rozdělení podle přístupu k výpočtu

1.5.1 Heuristické řešení

Heuristické řešení není výpočtově ani implementačně náročné jako přesné řešení. Při heuristickém řešení jsou tvary seřazeny podle velikosti a umisťovány do řádků. Řádkem se rozumí horizontální prostor na výchozí ploše, ze které jsou tvary následně vyřezávány. Řádky si lze představit jako polici na knihy, kde velikost každého řádku je dána prvním tvarem na řádku. Díky seřazení tvarů nemůže nastat, že následující tvar bude větší, jak tvar předchozí.

Umisťování tvarů probíhá zleva doprava, zdola nahoru. To znamená, že největší tvar je vždy umístěn v levém dolním rohu.

Tvary mohou být před umístěním otočeny ve vztahu k ostatním tvarům, nebo ve vztahu k základní desce. Jedná se o určitou formu heuristiky, jak dosáhnout lepšího výsledku, často je však výsledek opačný. Umisťování tvarů do řádků probíhá podle algoritmů známých z operačních systémů pro přidělení paměti procesu. Lze tedy použít jakýkoliv algoritmus pro přidělení paměti, nejčastěji se však používají tři algoritmy – next fit, first fit a best fit.

Nad rámec těchto algoritmů lze provést i algoritmy s vyšší optimalizací, kdy jsou vyhledávány volné prostory, do kterých jsou umisťovány další tvary. Takové algoritmy jsou například *Size alternating stack (SAS), Floor ceiling (FC)* nebo *Split fit (SF)*. Tyto algoritmy nejsou pouze řádkově orientované, ale při výpočtu se používají další metody dělení. Každé sadě tvarů může vyhovovat jiný algoritmus. Ve většině případů je ale nejlepším algoritmem best fit.

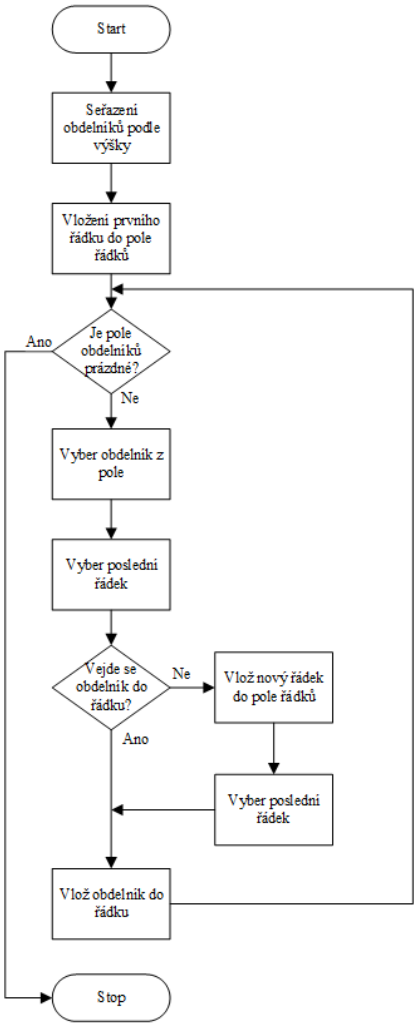
1.5.2 Přesné řešení

Přesné řešení je velice výpočtově i implementačně náročné. Při výpočtu je použito spoustu kombinací čísel. Využívá se lineárního programování, simplexové metody a metody branch and bound. Pro výsledný řezný plán je potřeba řešit polynomy n-tého stupně. Průměrný čas výpočtu v jazyce C++ se pohybuje kolem 60 s. Zmíněný výpočet byl prováděn na počítači s procesorem Intel Xeon s frekvencí 2,66 GHz a 8 GB paměti RAM. I přes vyšší přesnost se tento přístup nevyplatí použít z důvodu velké časové náročnosti. [23]

1.5.3 Next fit decreasing height

Jedná se o nejméně náročný algoritmus se zkratkou NFDH. Je to řádkově orientovaný algoritmus, ve kterém jsou tvary seřazeny sestupně podle výšky a následně umisťovány do řádků, kde jeho výška odpovídá prvnímu vloženému tvaru. Pokud je šířka tvaru větší než zbývající šířka řádku, je vytvořen nový řádek. Tvary se umisťují zleva doprava a řádky se vytváří zdola nahoru. Pořadí tvarů v řádcích odpovídá zásobníku, ze kterého jsou brány. Algoritmus se nevrací k předchozím řádkům, ale aktuální je vždy poslední řádek. [4]

Složitost algoritmu [5]: $O(n \cdot \log n)$

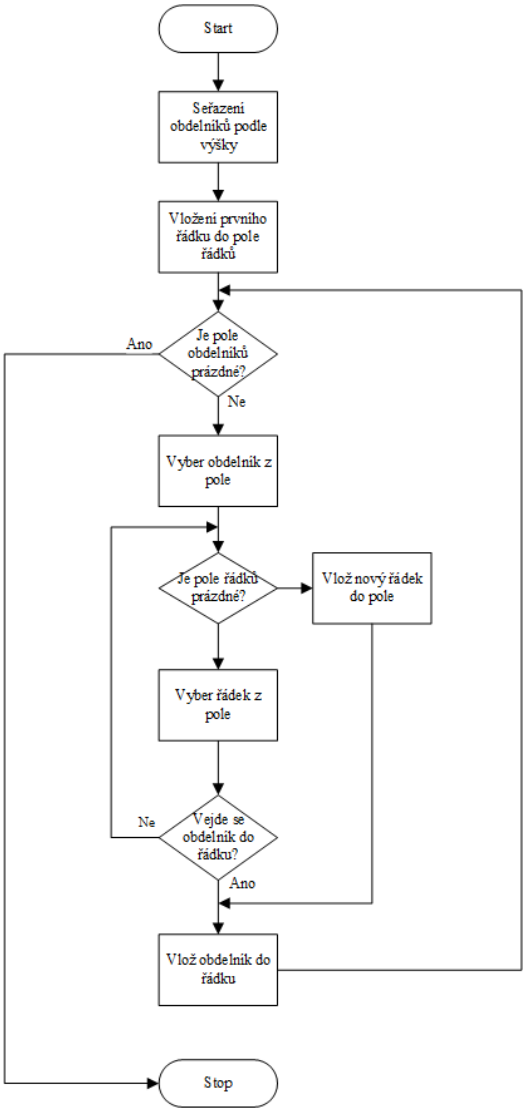


Obrázek 3- Vývojový diagram algoritmu next fit

1.5.4 First-fit decreasing height

Algoritmus nese zkratku FFDH. Stejně jako u algoritmu next-fit se jedná o řádkově orientovaný algoritmus, ve kterém jsou tvary seřazeny sestupně podle výšky a první řádek má výšku jako první vložený tvar. Z pole je vybrán první tvar, který se snažíme umístit do řádku. Od algoritmu next-fit se liší v tom, že tvar se snažíme umístit vždy již do existujících řádků. Řádek se tedy vybírá od prvního po poslední. Pokud se tvar do žádného řádku nevejde, je založen řádek nový. [4]

Složitost algoritmu [5]: $O(n \cdot \log n)$

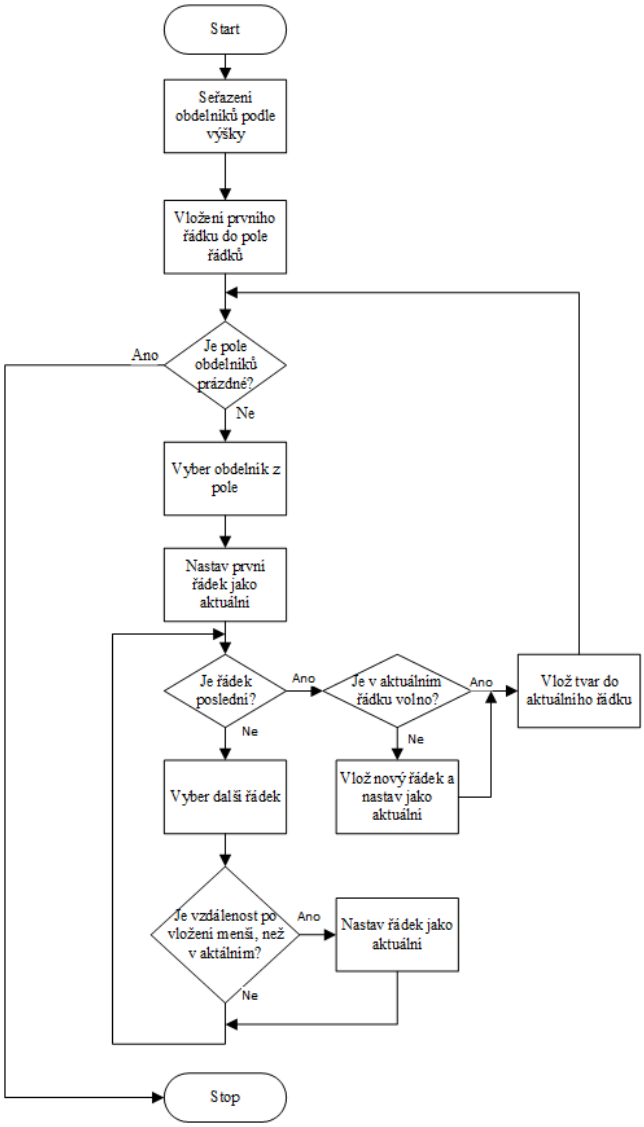


Obrázek 4- Vývojový diagram algoritmu first fit

1.5.5 Best-Fit decreasing height

Tento algoritmus nese zkratku BFDH a je podobný předešlým algoritmům. Nejprve jsou seřazeny tvary podle výšky od největšího po nejmenší, tvary jsou opět vybírány ze zásobníku a umisťovány do řádků. Od předešlých dvou algoritmů se *Best-Fit* liší v hledání volného místa. Jsou prohledány všechny řádky, ve kterých se zjišťuje volný prostor pro tvar. Z množiny je vybrán řádek, ve kterém bude odpad po umístění tvaru nejmenší.

Složitost algoritmu [5]: $O(n^2)$

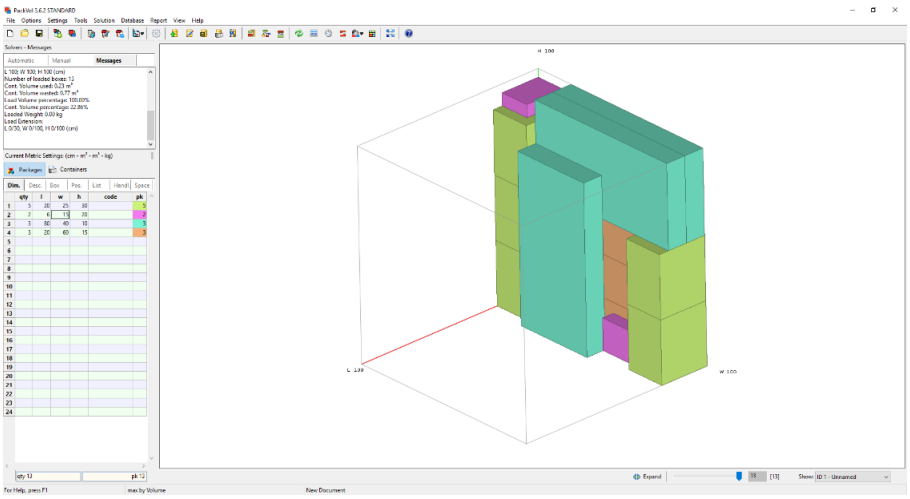


Obrázek 5- Vývojový diagram algoritmu best fit

1.6 Existující programy

1.6.1 PackVol

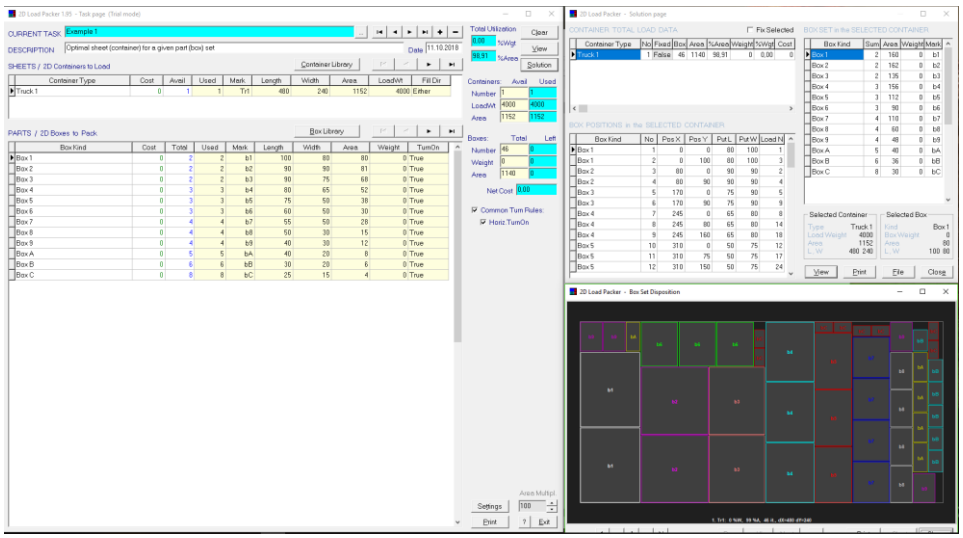
Program je určený pro 3D Bin packing problem. Je dostupný ve 3 verzích – Lite, Standart a Dynload. Jednotlivé verze se od sebe liší přístupem k boxům a ke kontejnerům. Lze mít k dispozici jeden, nebo více kontejnerů, do kterých budou boxy umisťovány. Ve verzi Dynload je navíc k dispozici nakládání boxů do kontejneru po krocích, což se hodí, pokud bude kamion vykládat zboží u více firem. Boxům lze přidat popisy, váhu a další parametry. Program je k dispozici pro operační systém Windows 7, 8, 8.1, 10 a Windows Server. Program je placený, ale lze stáhnout zkušební verzi na 14, nebo 30 dní. Délka zkušební verze se liší podle verze programu. [6]



Obrázek 6- Prostředí programu packVol 3.6.2 STANDART

1.6.2 Astrokettle 2D Load Packer

Program slouží k řešení 2D Bin packing problemu. K dispozici je i pro 3D Bin packing problem. Kromě základních rozměrů boxu program dovoluje přidání i jejich popisu. Stejně jako boxy, i kontejnery lze pojmenovávat, nebo čerpat a ukládat do databáze. Bohužel program má nepřehledné GUI a jeho licence je drahá, ale je k dispozici i ve zkušební verzi na 30 dní zdarma. Program je k dispozici na všechny operační systém Microsoft Windows. [7]

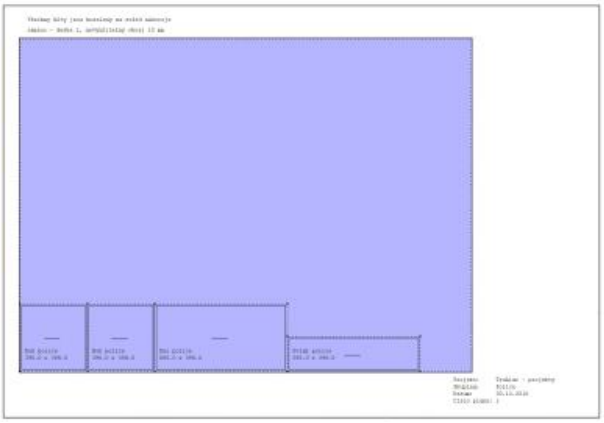


Obrázek 7- Prostředí programu Astrokettle 2D Load Packer 1.9.5

1.6.3 Truhlář

Truhlář je doplněk do programu SketchUp. Doplněk je určen především pro truhláře. Lze v něm vytvářet vlastní materiály s vlastním nastavením tloušťky, ceny za jeden metr, nebo dodavatele. Dále doplněk umožňuje tvorbu náhledového obrázku, celkovou spotřebu dřeva,

cenu za výrobek a v neposlední řadě také tvorbu řezného plánu. Při tvorbě řezného plánu jsou nepravidelné tvary převedeny do obdélníků, se kterými je následně počítáno. Bohužel doplněk nezobrazuje v řezném plánu řezy u nepravidelných tvarů, ale pouze obdélníky, které je ohraničují. Doplněk je k dispozici zdarma pro program SketchUp 6 a vyšší. [8]



Obrázek 8- Řezný plán v doplňku Truhlář [8]

2 PRAKTICKÁ ČÁST

Cílem bakalářské práce je vytvoření webové aplikace. Proto v následující kapitole je popis technologií a nástrojů potřebných pro tvorbu webové aplikace.

2.1 Použité technologie

2.1.1 Hypertext Markup Language

Značkovací jazyk HTML vytvořil Tim Berners-Lee když pracoval v CERNu jako součást projektu WWW. Jazyk HTML vycházel z jazyka SGML, který však byl příliš složitý. Cílem tedy bylo jazyk zjednodušit tak, aby byl přívětivější k tvůrcům webu. První verze jazyka byla vyvinuta v roce 1991.

Následoval vývoj jazyka, v roce 1999 vyšel standart jazyka HTML 4.01, který se používal dlouhou dobu. O rok později přišlo XHTML, což byla snaha skloubit jazyk HTML s jazykem XML. Toto vylepšení jazyka nepřineslo nic nového, jen omezení. Později se ukázalo, že je to slepá ulička a vývoj byl ukončen. Až v roce 2012 přišla první specifikace HTML 5. To přineslo spoustu nových funkcí, mezi největší změny patří podpora multimédií a geolokace.

Jazyk HTML zajišťuje vytvoření základní kostry webu. Je složen ze značek neboli tzv. tagů. Ty obalují text a tím určují význam. Značky jsou buď párové, nebo nepárové a mohou mít atributy určující jejich vlastnosti. [11] [24]

2.1.2 PHP

Skriptovací jazyk PHP vytvořil programátor Rasmus Lerdorf v roce 1994. O rok později byla vydána první verze tohoto jazyka s otevřeným zdrojovým kódem a disponovala jednoduchým parserem. Druhá verze jazyka nesla název PHP/FI, kde zkratka FI znamená Form Interpreter. Tento interpret uměl pracovat s prvky formulářů a umožňoval komunikaci s databázovým serverem. V roce 1997 programátoři Zeev Suraski a Andi Gutmans přepsali jádro jazyka, čímž vznikl PHP/FI 2. Následně vychází třetí verze jazyka, která nese i změnu jména. Nyní se jazyk jmenuje PHP: Hypertext Preprocessor.

V roce 2000 vychází další verze jazyka. Oproti předchozí verzi jazyk opět mění jádro, tentokrát je to Zend Engine 1.0. Byl přidán datový typ Boolean, podpora sériového portu COM na OS Windows a nově také podpora objektově orientovaného programování. O několik let později, v roce 2004 vychází PHP 5, které opět přináší řadu novinek. Mezi nejvýznamnější patří zabudovaná databáze SQLite, podpora výjimek pomocí bloku try-catch a vylepšení objektově orientovaného programování. To umožňuje v jazyce PHP uplatňovat programátorské techniky

z vyšších programovacích jazyků. Poté následovala šestá verze jazyka, která měla přinést podporu znaků Unicode. Nakonec tato verze nebyla nikdy vydána. [9] [11]

2.1.3 PHP 7.3

Momentálně nejnovější verze je PHP 7.3. Mezi nejvýznamnější vylepšení oproti předchozí verzi patří vylepšení rychlosti. Oproti PHP 5 je PHP 7 dvakrát rychlejší. Dále je již možné do metody, nebo funkce napsat typ návratové hodnoty. Ta se píše za argumenty před tělo metody, nebo funkce. [10]

2.1.4 CSS

Kaskádový styl neboli CSS je jazyk, ve kterém se nastavují zobrazovací pravidla pro elementy za jazyků HTML, XHTML a XML. První verze tohoto jazyka se objevila v roce 1996, kdy bylo rozhodnuto, že musí být oddělena data webové stránky od grafické podoby stránky, proto byl zaveden jazyk CSS. Ten si během svého vývoje prošel prudkým vývojem. Jazyk má dodnes problémy stejného vykreslení webové stránky v různých prohlížečích. Důvodem je odlišná implementace určitých elementů ze strany webových prohlížečů.

Nejnovější verze je CSS3 a jsou do ní přidávány stále nová vylepšení. Oproti svému předchůdci – CSS2, CSS3 podporuje zakulacené rohy u elementů, vržený stín, více jak jeden obrázek na pozadí, průhlednost elementů a přechody. [12]

2.1.5 Javascript

Javascript vznikl v roce 1992 ve společnosti Sun. Původně byl zamýšlen jako multiplatformní jazyk, který bude mít syntaxi podobnou s jazykem C, ale oproti tomu bude obsahovat věci navíc. Ve stejnou dobu se na webu řeší dynamický obsah webových stránek, který prozatím zajišťují Java applety. Javascript byl navržen za pouhé 3 týdny. Původně se měl jmenovat jinak, ale Sun trval na názvu Javascript, ten měl naznačovat, že se jedná o odlehčený jazyk Java, který zároveň vyvíjel. S Javou však nemá kromě syntaxe nic společného. Název je nešťastný a spousta lidí si ho dodnes spojuje s Javou. V roce 1996 byl Javascript implementován Microsoftem do Internetu Explorer a o dva roky později byl schválen jako standart ECMAscript.

Jedná se o interpretovaný jazyk, což znamená, že zdrojový kód je překládán za běhu na straně klienta. V tomto směru se liší od PHP, kde je kód překládán na straně serveru. Pro uchování dat používá jazyk pouze pět datových typů – Boolean, Number, String, Object a nově též Symbol. Jazyk je objektový a využívá funkcionální paradigma, což znamená, že do

proměnné lze uložit funkci, proto lze využít funkci jako atribut objektu, nebo návratovou hodnotu jiné funkce.

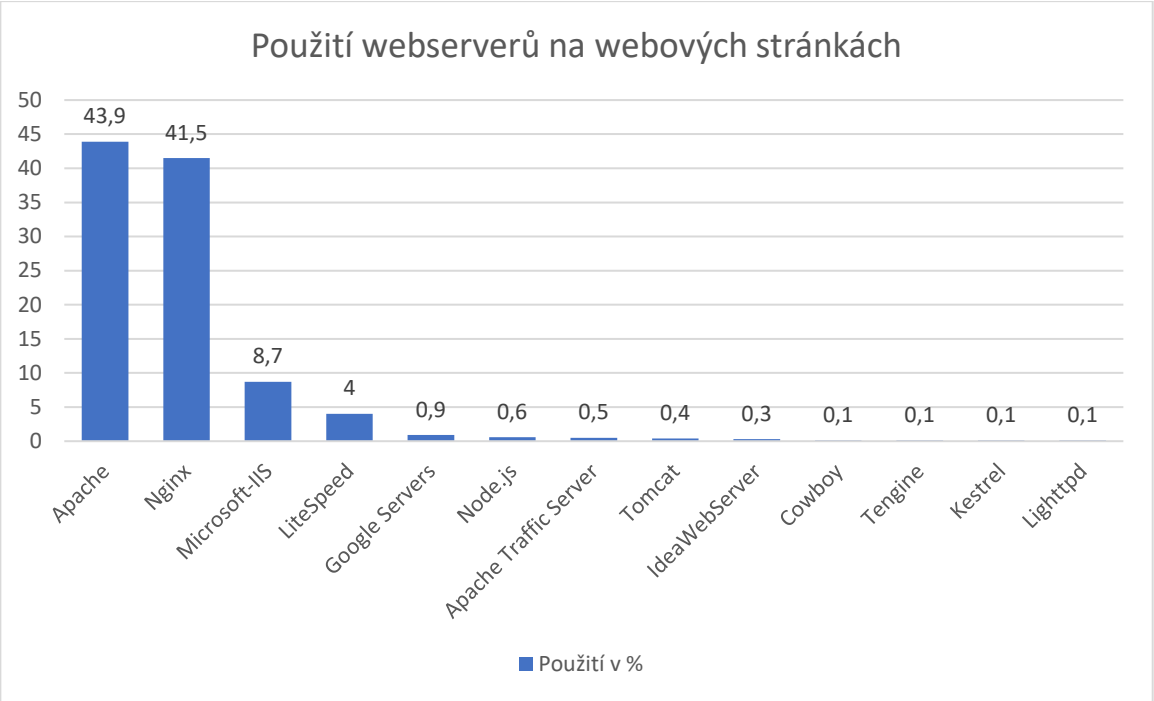
Jazyk bohužel trpí jednou velkou nevýhodou, kterou je zobrazení zdrojového kódu na straně klienta. Klient si tedy může zdrojový kód zobrazit a upravit dle libosti. To se nehodí například u validace polí, protože klient si může validovat cokoliv. Proto má validace v Javascriptu spíše informační charakter a na straně serveru musí být data ověřena znovu.

Javascript dnes používá většina webových aplikací pro dynamickou tvorbu obsahu. První, kdo začal Javascript hojně používat a stál za jeho rozšířením byl Google. Ten začal používat u svého emailového klienta službu AJAX založenou na Javascriptu v době, kdy se ostatní báli tuto technologii používat. Dnes Javascript používají kancelářské balíky pro svůj online editor – Google Dokumenty a Microsoft Office. Dále Javascript používá Facebook pro aktualizaci zdi. Javascript se zasloužil o vytlačení Flashe z webových stránek. Aplikace dříve běžící ve Flashi dnes běží v Javascriptu. Výhodou je vestavěná implementace v prohlížeči, uživatel nemusí nic dodatečně instalovat. [13]

2.1.6 Apache

Apache je nejrozšířenější webový server a je dostupný zdarma. Jeho název je odvozen od původních obyvatel Ameriky – Apačů. Proto má v logu ptačí pero – symbol Apačů. Vznikl v roce 1993 na univerzitě v Illinois pod názvem NCSA HTTPd. O rok později opustil vývojářský tým jeho hlavní programátor, tím se vývoj zpomalil a v roce 1998 se zastavil úplně. Mezitím se však Apache používal na webserverech a jejich správci mu dodávali vlastní záplaty. To vedlo k založení emailové konference, která tyto záplaty začala sbírat a distribuovat. V roce 1995 byla vydána první veřejná verze. Následovalo založení vývojářské skupiny Apache Group, která se o správu Apache stará dodnes. [14]

Od roku 1996 je Apache nejpoužívanějším webserverem. Aktuální procentuální použití webserverů je zobrazeno na obrázku 9.



Obrázek 9- Graf použití webserverů na webových stránkách [15]

2.1.7 Bootstrap

Bootstrap je webový framework, který vyvinul Mark Otto a Jacob Thornton původně jako interní framework pro Twitter. Původně nesl jméno Twitter Blueprint a to až do akce Hack Week, kde se přejmenoval na Bootstrap. Na této akci se Bootstrap těšil velké oblibě hlavně mezi vývojáři. Ti i s minimálními zkušenostmi dokázali díky Bootrapu vytvořit pěkné webové stránky. Od verze 3 je framework plně responzivní a díky tomu se jedná o jeden z nejpoužívanějších webových frameworků. Aktuální verze Bootstrapu je 4.3. [16]

Bootstrap se hodí především pro tvorbu méně rozsáhlých webů a aplikací, u kterých není plánována nějaká zásadní změna vzhledu oproti originálu. Pokud má programátor v plánu větší úpravy vzhledu – firemní identita, nebo tvorba vlastních komponent, doporučuji jiný framework, např. Zend, nebo Foundation.

2.1.8 Konva

Konva je HTML5 Javascriptová knihovna pro tvorbu 2D tvarů a obrazců umožňující animaci, přechody, vnořování, filtrování a vrstvení tvarů. Dále je lze transformovat a přidávat na ně události a obslužné rutiny. Knihovna disponuje dobrou dokumentací, kde se nachází popis komponent a spousta ukázkových příkladů. Konva je výborně optimalizována – i při větším počtu tvarů se výsledek bude zobrazovat rychle a korektně. Konva vznikla oddělením od

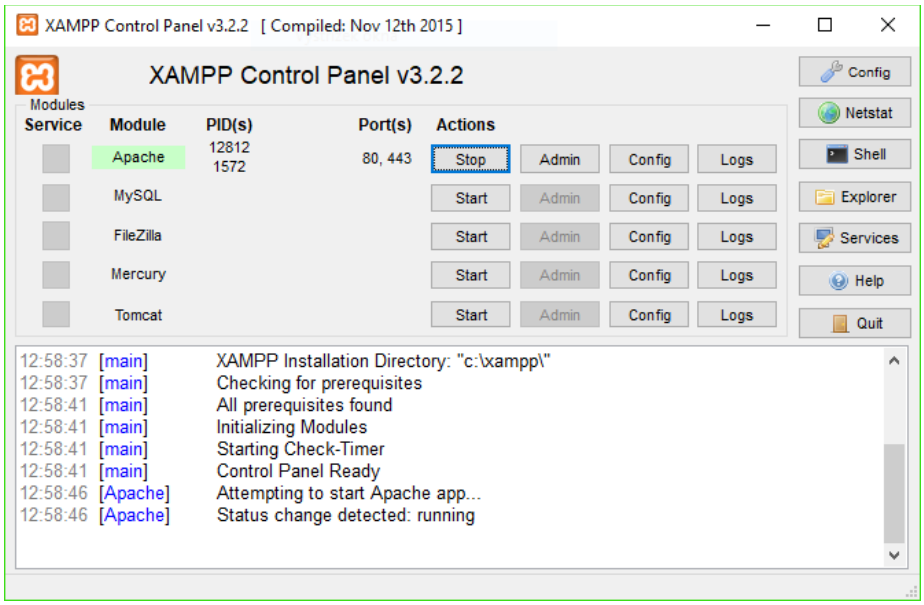
knihovny KineticJS na serveru GitHub.com, která se od roku 2016 již nevyvíjí. Naopak Konva se stále aktivně vyvíjí. [17]

2.1.9 Html2canvas

Jedná se o Javascriptovou knihovnu, která umožňuje vytvoření snímku celé stránky, nebo její části. Knihovna vytváří snímky podle standartu DOM. Je tedy možné, že v jednotlivých prohlížečích se snímky mohou nepatrně lišit. I přesto, že knihovna podporuje většinu CSS vlastností, bohužel nepodporuje všechny. Mezi nepodporované vlastnosti patří například border-image, box-shadow, nebo zoom. Dále knihovna nepodporuje vytvoření snímku Flashe, nebo Java appletů. Naopak knihovna podporuje všechny běžně dostupné prohlížeče. [18]

2.1.10 XAMPP

XAMPP je přednastavený balík služeb obsahující nejpoužívanější technologie pro tvorbu webových aplikací. Balík obsahuje webový server Apache, relační databázi MariaDB, skriptovací jazyk PHP a interpretovaný jazyk Perl. Aby byl XAMPP pro vývojáře co nejjednodušší, jsou všechny služby ve výchozím stavu již nakonfigurovány. Všechny tyto technologie jsou dostupné zdarma, stejně jako samotný XAMPP. Ten je k dispozici na platformy Windows, Linux a OS X. [11] [19]



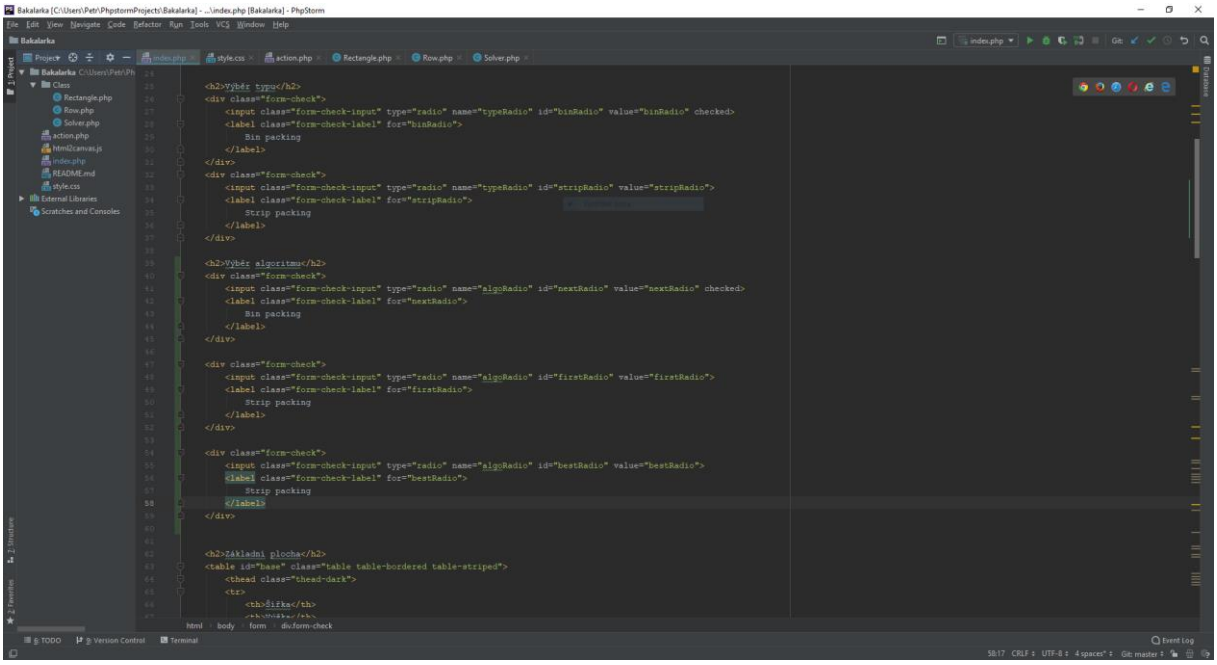
Obrázek 10- Řídící panel XAMPP

2.1.11 Vývojové prostředí PhpStorm

Pro vývoj webové aplikace jsem si vybral IDE PhpStorm od české společnosti JetBrains. Jedná se o integrované webové prostředí primárně určené pro psaní webových aplikací v jazyce

PHP. Mimo to ale podporuje i psaní kódu v HTML, CSS a Javascriptu. Dále podporuje nejpoužívanější frameworky pro webové aplikace jako je například Symfony, Drupal, WordPress, nebo Zend. Projekty je možné snadno verzovat díky vestavěné podpoře pro systém správy verzí.

Vývojové prostředí je velice jednoduché a přehledné a umožňuje v něm uživatelům snadnou orientaci. Spolu s vestavěným nástrojem pro debuggování – Xdebug a klávesovými zkratkami umožňuje rychlé a efektivní psaní kódu. [20]



Obrázek 11- IDE PhpStorm 2018.3.4

2.1.12 Git

Systém pro správu verzí je dnes již nedílnou součástí vývoje každé webové aplikace. Pomocí verzování je uživatel, ale i celý tým schopen svůj zdrojový kód pravidelně zálohovat, upravovat, vytvářet různé verze a vracet se v historii projektu. Správa verzí se hodí zejména pro týmové projekty, ale nevylučuje se ani pro jednotlivce. Projekty jsou uloženy v repositářích, které jsou uloženy na cloudu. To nese výhodu, že uživatel k nim může přistupovat odkudkoliv ze světa.

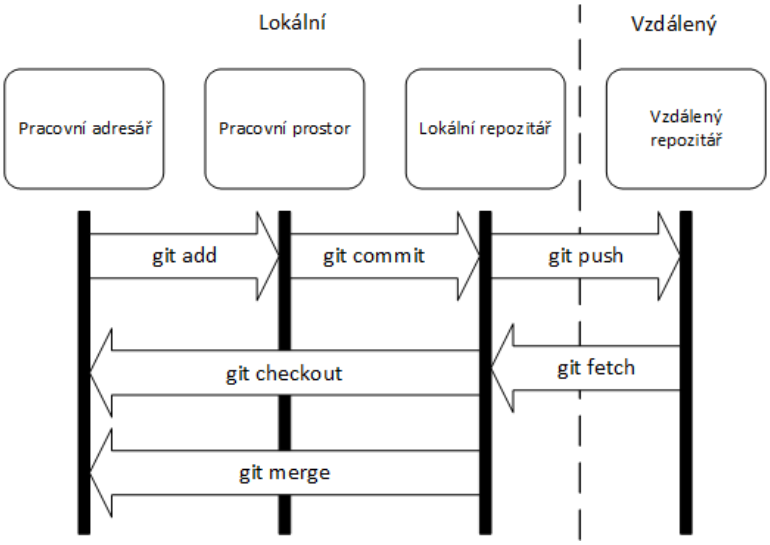
Systémy pro správu verzí lze rozdělit do dvou kategorií. První kategorii tvoří centralizované systémy. Jak už z názvu vypovídá, tak celý projekt je uložen na centrálním úložišti, odkud si ho programátoři stahují a provádí na něm změny. Při zpětném nahrávání server kontroluje

změny – soubory, na kterých pracovalo více programátorů zároveň. Mezi systémy pro správu verzí patří Apache Subversion.

Druhá kategorie se nazývá distribuované systémy. Princip těchto systémů spočívá v tom, že programátor si stáhne celý repozitář, čímž si vytvoří lokální kopii, na které potom pracuje. V revizích nejsou uchovávány všechny soubory, ale jen rozdílné. Verze souborů potom lze slučovat, nebo zamykat. [11]

Já jsem si vybral právě distribuovaný systém správy verzí, přesněji GitHub založený na Gitu. Zvolil jsem ho na základě předchozích zkušeností, velké komunity a vestavěnému nástroji VCS v PhpStormu pro správu verzí.

Za vytvořením Gitu stojí linuxoví vývojáři v čele s otcem Linuxu Linusem Torvaldsem. K tomuto kroku je vedla správa verzí linuxového jádra. To bylo původně spravováno pomocí Bitkeeperu. Bohužel tento systém byl později zpoplatněn, což vedlo k vytvoření Gitu. Úkolem Gitu tedy je zajistit správu verzí nad velkými projekty a zároveň programátorovi zajistit maximální pohodlí. [21]



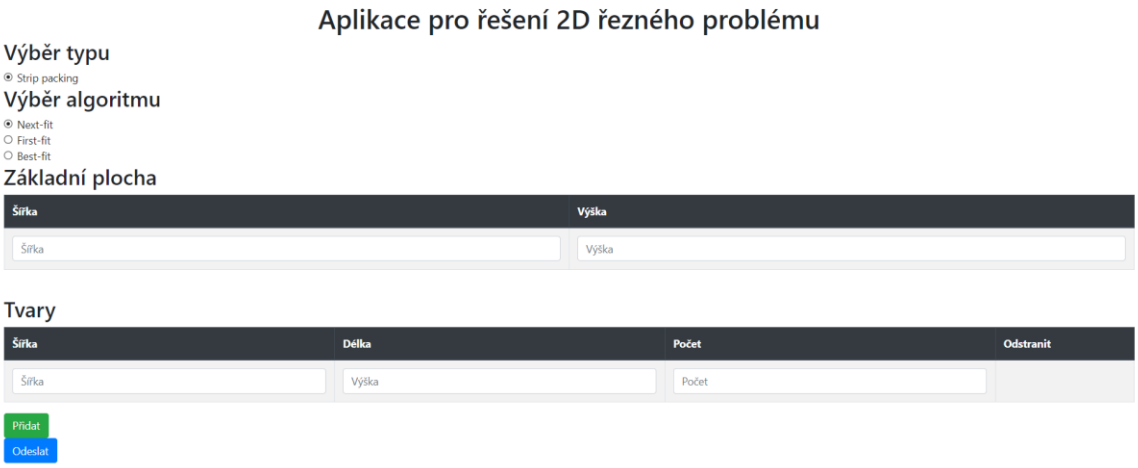
Obrázek 12- Vrstvy Gitu

Prostor Gitu lze rozdělit na lokální a vzdálený. V lokálním adresáři se nachází *Pracovní adresář*. Jedná se o složku, ve které pracujeme. Neexistuje žádné pravidlo, že všechny soubory v této složce musí být i na repozitáři, nebo naopak. Další složkou je *Pracovní prostor*. Lze si ji představit jako virtuální složku a vše, co složka obsahuje je při commitu přeneseno do repozitáře. V lokálním repozitáři je uložena historie projektu, tj. všechny verze projektu. Poslední složkou je *Vzdálený repozitář*. Na ten se nahrávají lokální repozitáře a slouží

především pro ostatní uživatele, nebo další pracovníky. Odtud si lze repozitář stáhnout, prohlížet, nebo klonovat. Mezi nejznámější vzdálené repozitáře patří GitHub a Gitlab. [21]

2.2 Grafické uživatelské rozhraní

Při tvorbě GUI jsem vycházel z trendu, že stále více návštěv na webu se uskutečňuje z mobilních zařízeních. Proto jsem se rozhodl, že webová aplikace bude responzivní. Dalším důvodem je, že vyhledávače upřednostňují responzivní webové stránky nad neresponzivními tím, že jim přidělují vyšší rank. Vybral jsem proto framework Bootstrap, který zaručí responzivitu webové aplikace pouhým přidáním knihoven do hlavičky souboru a přidáním třídy do elementu podle dokumentace Bootstrapu. Předdefinované elementy v Bootstrapu mají moderní vzhled, dostatečnou velikost a jsou responzivní. To zaručí komfort uživatele při používání webové aplikace. Při vytváření GUI jsem se držel doporučením podle zdroje [1].



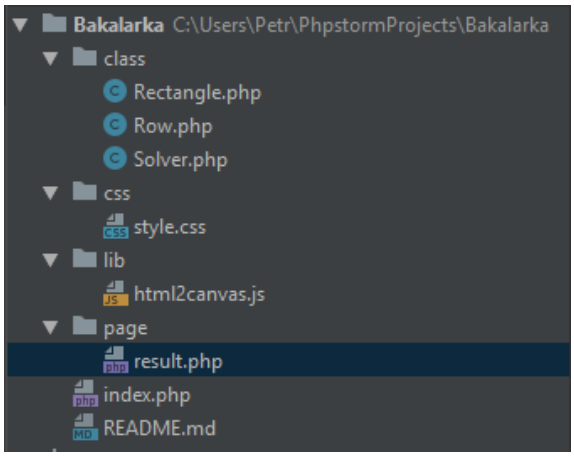
Obrázek 13- GUI webové aplikace

Grafické uživatelské rozhraní disponuje výběrem algoritmu, který bude použit při výpočtu. K výběru má uživatel tři základní algoritmy. Pro přepínání mezi algoritmy je použita komponenta radio button. Do polí základní plochy uživatel vyplní šířku a výšku plochy, na kterou se budou tvary umisťovat.

Pole tvarů slouží pro vyplnění informací o tvarech. Je požadována šířka, výška a počet výskytu tvaru. Další tvary lze přidat pomocí tlačítka *Přidej*. Pokud potřebuje uživatel tvar z pole tvarů odstranit, klikne na tlačítko *Odebrat*.

2.3 Adresářová struktura

V adresářové struktuře jsem se rozhodl oddělit jednotlivé soubory do adresářů pro vyšší přehlednost. Oddělil jsem soubory tříd od ostatních souborů, jako jsou knihovny a soubory kaskádových stylů.



Obrázek 14- Adresářová struktura aplikace

- class – Adresář class obsahuje třídy jazyka PHP, které se používají pro výpočet nářezového plánu. Tyto třídy zapouzdřují jednotlivé uživatelem zadané údaje, které třída Solver využívá.

- css – V tomto adresáři je umístěn soubor kaskádového stylu.

- lib – Zde se nachází Javascriptová knihovna sloužící k vytvoření obrázku.

- page – V adresáři *page* se nachází soubor *result.php*. V tomto souboru se počítá nářezový plán, který je zde následně zobrazen.

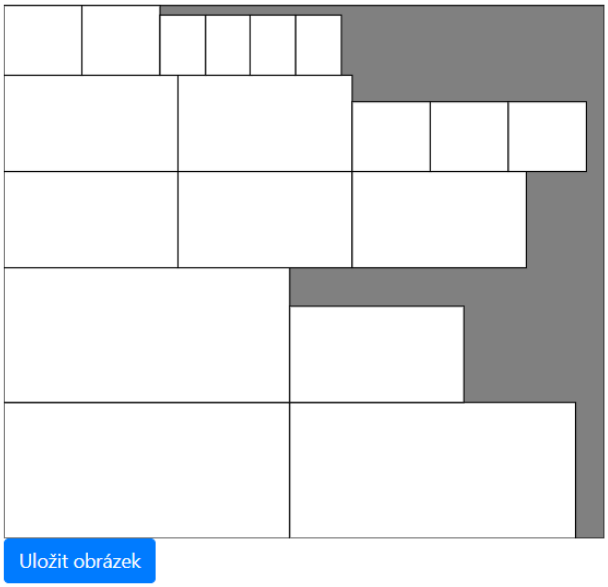
2.4 Výstup aplikace

Výstupem aplikace je obrázek a textová data. Obrázek reprezentuje data – nářezový plán, který je výstupem třídy *Solver*. Nářezový plán je ve formátu JSON. Pro tento formát jsem se rozhodl, protože je přímo určen pro výměnu dat, snadno se vytváří a snadno se zpětně parsuje. Tento text je uložen ve skrytém poli. Pomocí Javascriptu je JSON zpětně parsován a pomocí grafické knihovny Konva je interpretován do podoby obrázku.

Obrázek jsem styloval do barev, ve kterých se nachází nářezové plány v odborné literatuře – podklad je tmavě šedé barvy a obdélníky jsou bílé barvy s černým obrysem.

Při vykreslování obrázku jsem řešil problém v podobě umisťování tvarů na základnu. Zatímco v literatuře se tvary umisťují od levého dolního rohu a jsou tomu uzpůsobeny všechny algoritmy, grafický canvas vykresluje tvary od levého horního rohu. Obrázek byl tedy svisle překlopen. Bohužel grafická knihovna Konva, kterou jsem použil nepodporuje překlápění, či otáčení plátna. Obrázek jsem proto překlopil pomocí kaskádových stylů.

Při ukládání obrázku jsem bohužel zjistil, že obrázek se uloží v původní, nepřeklopené podobě. Tuto chybu jsem vyřešil pomocí Javascriptové knihovny *html2canvas*. Knihovna je schopna vytvořit snímek jakékoliv části webové stránky. Proto jsem výsledný obrázek umístil do elementu div. Knihovna poté vytvoří vybraného div elementu ve správné, již nepřeklopené pozici po kliknutí na tlačítko a uživateli se zároveň obrázek uloží.



Obrázek 15 - Výsledek řezného plánu v grafické formě

Dalším výstupem aplikace je textová podoba řezného plánu. Ta je dána šířkou a výškou obdélníka a jeho pozicí v kartézské soustavě souřadnic, tedy pozicí x a y.

```
šířka: 95       výška: 45       x: 0    y: 0
šířka: 95       výška: 45       x: 95   y: 0
šířka: 95       výška: 45       x: 0    y: 45
šířka: 58       výška: 32       x: 95   y: 45
šířka: 58       výška: 32       x: 0    y: 90
šířka: 58       výška: 32       x: 58   y: 90
šířka: 58       výška: 32       x: 116  y: 90
šířka: 58       výška: 32       x: 0    y: 122
šířka: 58       výška: 32       x: 58   y: 122
šířka: 26       výška: 23       x: 116  y: 122
šířka: 26       výška: 23       x: 142  y: 122
šířka: 26       výška: 23       x: 168  y: 122
šířka: 26       výška: 23       x: 0    y: 154
šířka: 26       výška: 23       x: 26   y: 154
šířka: 15       výška: 20       x: 52   y: 154
šířka: 15       výška: 20       x: 67   y: 154
šířka: 15       výška: 20       x: 82   y: 154
šířka: 15       výška: 20       x: 97   y: 154
```

Obrázek 16- Textová forma řezného plánu

2.5 Porovnání výstupu

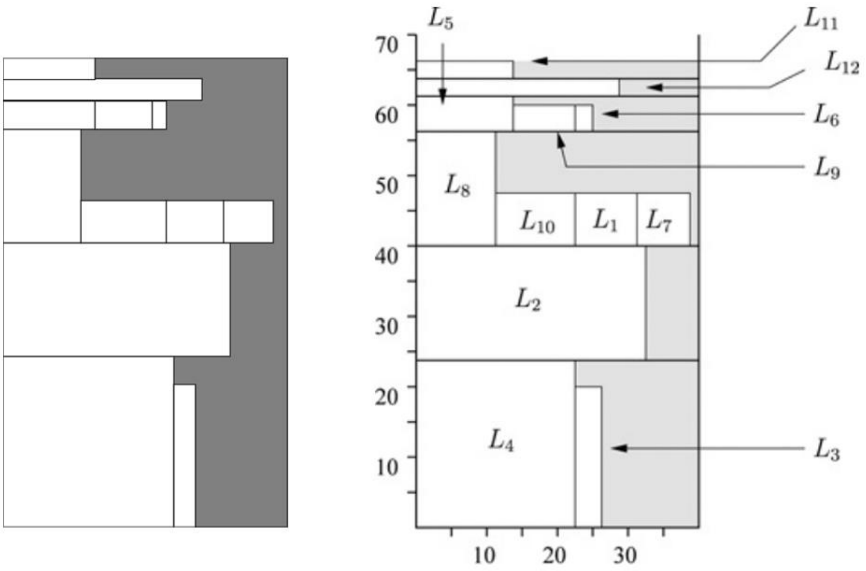
Pro ověření správnosti výstupu aplikace jsem použil data z tabulky 1. Typem řezného problému je strip packing problem, základní deska měla šířku 40 a správnost byla testována na třech základních algoritmech.

Tabulka 1- Testovací hodnoty pro výpočet [4]

	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}	L_{11}	L_{12}
$h(L_i)$	6	16	20	24	4	4	6	16	4	6	3	3
$w(L_i)$	8	32	3	24	13	2	7	11	8	12	13	28

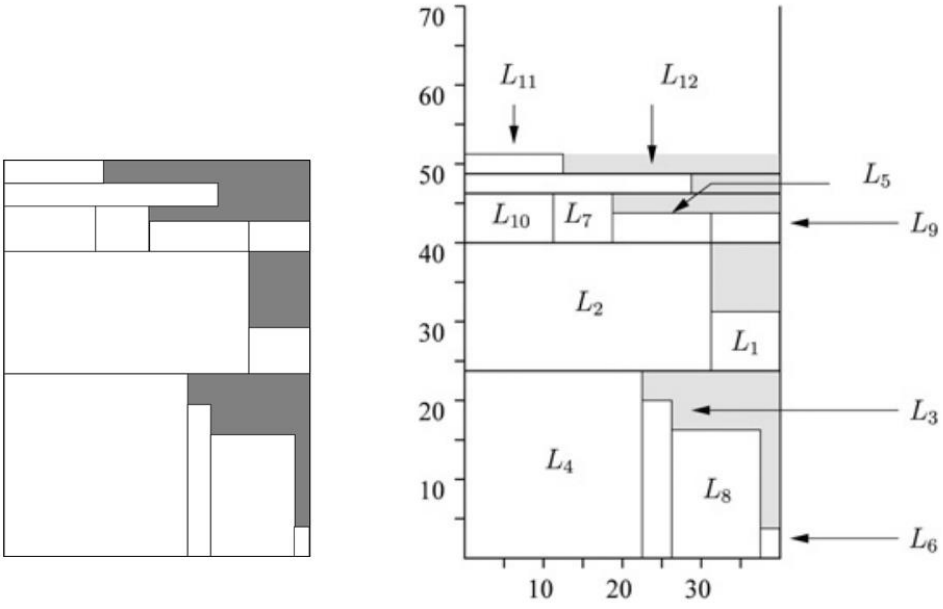
- **L_i** – označení obdélníku
- **h** – výška obdélníku
- **w** – šířka obdélníku

2.5.1 Porovnání výstupu next fit



Obrázek 17- Porovnání výstupu pro algoritmus next fit, vlevo můj nářezový plán, vpravo [4]

2.5.2 Porovnání výstupu best fit a first fit



Obrázek 18- Porovnání výstupu pro algoritmus best fit a first fit, vlevo můj nářezový plán, vpravo [4]

2.5.3 Porovnání heuristického a přesného algoritmu

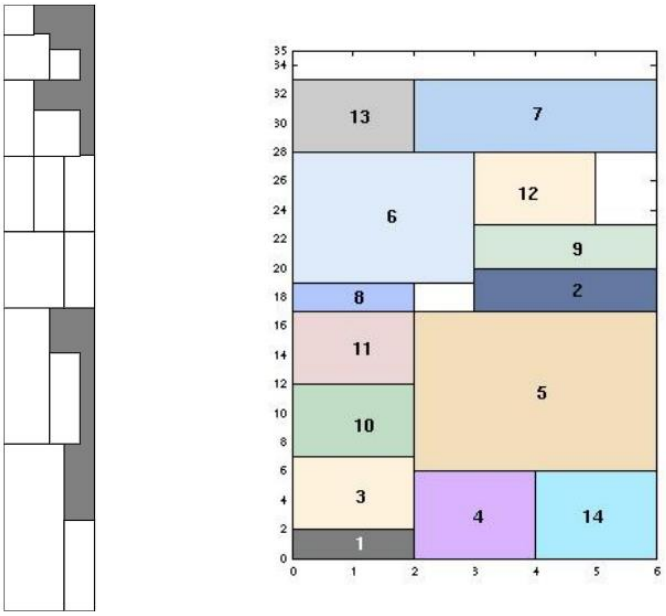
Při porovnání jsem porovnával heuristický přístup s exaktním. Pro nářezový plán tvořený pomocí heuristické metody jsem použil algoritmus best fit. Srovnával jsem ho s již hotovým

nářezovým plánem, který byl vypočítán podle exaktního algoritmu, jehož rozměry tvarů jsou uvedeny v tabulce 2. Bohužel exaktní algoritmus využívá otočení, zatímco můj heuristický nikoliv. Šířka základní desky byla 6 jednotek. Využití materiálu mého řezného plánu využívajícího heuristickou metodu bylo 80 %, využití materiálu řezného plánu využívajícího přesnou metodu bylo 97 %. [22]

Tabulka 2 - Rozměry tvarů pro porovnání [22]

	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}	L_{11}	L_{12}	L_{13}	L_{14}
$h(L_i)$	2	3	5	6	11	9	5	2	3	5	5	5	5	6
$w(L_i)$	2	3	2	2	4	3	4	2	3	2	2	2	2	2

- **L_i** – označení obdélníku
- **h** – výška obdélníku
- **w** – šířka obdélníku



Obrázek 19 - Porovnání heuristické a přesné metody, vlevo můj nářezový plán, vpravo [22]

2.5.4 Výsledek porovnání

Při porovnání jsem zjistil, že moje aplikace počítá řezný plán zcela korektně – tvary jsou umístěny na stejných místech a ve stejném pořadí. Při bližším prozkoumání lze ale zjistit, že obrázky řezných plánů nejsou zcela shodné, moje aplikace zobrazuje výsledný řezný plán daleko přesněji. Důvodem nejspíše je že můj obrázek je generován pomocí grafické knihovny a druhý obrázek byl kreslen autorem pouze jako přibližná ilustrace.

Při porovnání heuristické a exaktní metody jsem zjistil, že tvorba nářezového plánu pomocí heuristické metody je dostačující a výpočet podle přesné metody se nevyplatí z důvodu časové náročnosti. I když oba algoritmy nebyly zcela srovnatelné, využitelnost materiálu se lišila o pouhých 17 %.

ZÁVĚR

Cílem bakalářské práce bylo navrhnout a implementovat webovou aplikaci pro řešení řezného problému, což bylo splněno. Vytvořená webová aplikace umožnuje výpočet nářezového plánu na základě uživatelsky zadaných dat. Výsledný nářezový plán je dostupný v textové i obrazové podobě.

Před popisem implementace a návrhu aplikace byla popsána problematika řezného problému, kde byl uvedeno rozdělení a popis základních algoritmů jak ve slovní podobě, tak pomocí vývojových diagramů. Před návrhem aplikace byl proveden průzkum již existujících programů pro řezný problém.

Následně byla navržena a implementována webová aplikace, ve které byly vybrány, popsány a implementovány technologie s ohledem na aktuální trendy. Ve finále byly výsledné nářezové plány z aplikace porovnány s jinými nářezovými plány a jiným přístupem k výpočtu.

Tato bakalářská práce byla mojí první prací většího formátu. S jejím výsledkem jsem velice spokojen. Díky tomu jsem nabil nových poznatků ohledně webových technologií, jejich aktuálních trendů a objevil problematiku řezného problému. Ta se mi velice zalíbila a v budoucnu bych aplikaci rád rozšířil o další metody a algoritmy. Problematice bych se rád věnoval i v budoucnu, nejlépe v diplomové práci, kde bych se tomuto tématu věnoval hlouběji.

POUŽITÁ LITERATURA

[1] CHINNATHAMBI, Kirupa. *Learning React: a hands-on guide to building maintainable, high-performing web application user interfaces using the React javascript library.* Hoboken, NJ: Pearson Education, 2016. ISBN 978-0134546315.

[2] HÁJEK, Petr. *Problematika nářezových plánů*. Brno, 2006. Diplomová práce. Masarykova univerzita. Vedoucí práce RNDr. Jaroslav Pelikán, Dr.

[3] AVDZHIEVA, Ana, Todor BALABANOV, Georgi EVTIMOV, Detelina KIROVA, Hristo KOSTADINOV, Tsvetomir TSACHEV, Stela ZHELEZOVA a Nadia ZLATEVA. *Optimal Cutting Problem* [online]. 2016, 49-61.

[4] NTENE, N. a J.H. VAN VUUREN. A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem. In: *Discrete Optimization*. 2009, **6**(2), s. 174-188. DOI: 10.1016/j.disopt.2008.11.002. ISSN 15725286. Dostupné také z: https://linkinghub.elsevier.com/retrieve/pii/S1572528608000844

[5] Survey on two-dimensional packing. *ARC project* [online]. Velká Británie, Německo [cit. 2019-04-08]. Dostupné z: http://cgi.csc.liv.ac.uk/~epa/surveyhtml.html

[6] ANGELUCCI, Antimo. *PackVol: Container Loading Optimization Software* [online]. Itálie, 2019 [cit. 2019-03-31].

[7] YURCHENKO, Nikita. *2D Load Packer* [online]. 2019 [cit. 2019-03-31]. Dostupné z: http://www.astrokettle.com/pr2dlp.html

[8] MIKAČ, Ivo. *Nástavba pro kreslení nábytku* [online]. 2016 [cit. 2019-04-01]. Dostupné z: http://www.mikac.cz/sketchup-truh-nastavba.html

[9] PHP History. *NuSphere* [online]. 2000–2019 [cit. 2019-03-23]. Dostupné z: http://www.nusphere.com/php/php_history.htm

[10] HUJER, Martin. Jaké novinky přinese PHP 7. *Zdroják.cz* [online]. Devel.cz Lab, 2015 [cit. 2019-03-23]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/jake-novinky-prinese-php-7/

[11] ČEGAN, Lukáš. *Vývoj webových aplikací v PHP a NetBeans*. Pardubice: Univerzita Pardubice, 2015. ISBN 978-80-7395-858-9.

[12] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: Názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

[13] ČÁPKA, David. Lekce 1 - Úvod do JavaScriptu. *ITnetwork.cz* [online]. Praha, 2016–2018 [cit. 2019-03-27]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

[14] Apache HTTP server. *Banan.cz: Webhosting* [online]. 2019 [cit. 2019-05-06]. Dostupné z: https://www.banan.cz/napoveda/slovnicek-pojmu/apache-http-server

[15] Usage of web servers. *W3 Techs: Web Technology Surveys* [online]. Q-Success, ©2009-2019 [cit. 2019-03-29]. Dostupné z: https://w3techs.com/technologies/overview/web_server/all

[16] About. *Bootstrap* [online]. 2019 [cit. 2019-03-29]. Dostupné z: https://getbootstrap.com/docs/3.3/about/

[17] Documentation. *KONVA* [online]. ©2019 [cit. 2019-03-23]. Dostupné z: https://konvajs.org/docs/

[18] VON HERTZEN, Niklas. About. *Html2canvas* [online]. 2019 [cit. 2019-05-08]. Dostupné z: http://html2canvas.hertzen.com/documentation

[19] About. *Apache Friends* [online]. ©2019 [cit. 2019-03-23]. Dostupné z: https://www.apachefriends.org/about.html

[20] PhpStorm. *Jetbrains* [online]. ©2000-2019 [cit. 2019-04-13]. Dostupné z: https://www.jetbrains.com/phpstorm/

[21] VALKOVIČ, Patrik. *Lekce 1 - Git - Historie a principy* [online]. Praha, 2014 [cit. 2019-04-12]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/software/git/git-tutorial-historie-a-principy

[22] A Comparative Study of Exact Algorithms for the Two Dimensional Strip Packing Problem. In: BEKRAR, Abdelghani, Imed KACEM a Chengbin CHU. *Journal of Industrial and Systems Engineering*. 2007, s. 151-170. ISSN 1735-8272.

[23] DELORME, Maxence, Manuel IORI a Silvano MARTELLO. Bin packing and cutting stock problems: Mathematical models and exact algorithms. In: *European Journal of Operational Research*. 2016, **255**(1), s. 1-20. DOI: 10.1016/j.ejor.2016.04.030. ISSN 03772217. Dostupné také z: https://linkinghub.elsevier.com/retrieve/pii/S0377221716302491

[24] A Brief History of HTML. *Web Design & Development I: Student Version* [online]. University of Washington, ©2005-2019 [cit. 2019-03-18]. Dostupné z: https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html