

go beyond  
e-banking of today

let the customer  
take control  
of his finances

# Idea

Today's e-banking websites are not very interactive, you have a lot of functionality which is spread over pages, making sense of one's transactions is quite hard.

Create a web application that makes the customer feel in control of his/her finances, make it a great experience by making the common and important tasks as easy as possible.

Your web application could help with...

- ... budgeting
- ... achieving saving goals
- ... common tasks like transferring money, making them dead simple
- ... making your transactions accessible in task-oriented ways
- ... make the web application accessible with a multilevel security model
- ... enabling mobile access

Let the customer take control with ease

# The dashboard shows the most important information and makes the most common task directly accessible

Today's e-banking site doesn't give a good overview on what's happening in your accounts, you don't see much more than your account balance.

Identify the most important information and display it directly on the dashboard, this might include balances of different accounts and movements in the main account. Furthermore the most important tasks should be directly accessible there (make payments, pay e-bills, transfer money between accounts). The dashboard should also show advanced features like setting saving goals and budgeting.

**UBS Dashboard** | [Accounts](#) | [Budget](#) | [Pay](#) | [Invest](#) | [Services](#) | [Settings](#) | [help](#)

**Balance**

<a href="#">Main</a>	7205 CHF
<a href="#">Savings</a>	19205 CHF
<a href="#">Taxes</a>	3400 CHF

**Investment**

<a href="#">Futura Fonds</a>	24'000 CHF
<a href="#">Obligations</a>	6'000 CHF

**Do**

- [Make payment](#)
- [Request payment](#)
- [Pay bills](#)
- [Transfer Money](#)

**Recent changes on main account**

				Balance
Dec 3rd	+250 CHF	from CC Corp	Refund	7205 CHF
Nov 28th	-1800 CHF	to Rent Corp	Rent	9005 CHF
Nov 28th	-1800 CHF	to Rent Corp	Rent	9005 CHF
Nov 28th	-1800 CHF	to Rent Corp	Rent	9005 CHF

**Your saving goals**

- [New Macbook Pro](#)  800 CHF left
- [Bike](#)  500 CHF left

**Your budget**

[December](#)  3400 CHF / 6450 CHF

Warning [Phone Utility](#) is 20% over budget

give an overview on all accounts

primary tasks can be accessed directly

show what's happening in your main account

Show the saving goals and relevant budget information

# Taking control means budgeting

Today maintaining control over a budget takes a lot of work, but it doesn't have to.

1. Let customers create budget accounts like *utility expenses* or *eating out*.
2. The user can then allocate transactions to those accounts.
3. Once a company has been associated with an account, the categorization can be automated, the customer only has to categorize a few transactions and he'll have a nice overview on what he's spending money on.

The image shows two screenshots of the UBS banking interface. The top screenshot displays the 'Main Account' with a balance of 7205 CHF and a list of transactions from December, November, and September. A red dashed box highlights the category dropdown menu for a transaction, showing options like 'CC', 'Rent', 'Utility', 'Gifts', 'Phone', 'Xtra', and '+ add category'. The bottom screenshot shows the 'Budget for current month (December)' page. It features a progress bar for the month (29% spent, 5460 CHF left) and a list of budget categories: Rent (0 CHF of 1800 CHF), Food / Groceries (240 CHF of 1050 CHF), Phone / Utility (570 CHF of 460 CHF, 20% over budget), and Gifts (240 CHF of 1050 CHF). A pie chart titled 'Money spent this month' shows the distribution: 28% Utility, 10% Gifts, and 18% Utility. Red arrows point from the text labels below to the corresponding elements in the screenshots.

assigning transactions to budget accounts

show the customer where he can spend money and where he is over budget

show the customer what he is spending money on

# Taking control means setting goals

Often you have to put away money for things you want to be able to afford in the future, but how can he keep track?

1. Let the customer define saving goals.
2. Once defined he can allocate money to those saving goals manually or by setting up a saving plan, which puts a certain amount of money towards the saving goal.

The screenshot displays the UBS dashboard interface. At the top, the UBS logo is on the left, and navigation links for Dashboard, Accounts, Budget, Pay, Invest, Services, and Settings are in the center. A help link is on the right. The main content is divided into two sections: 'Your saving goals' and 'Overview'. Under 'Your saving goals', there are two goal cards. The first is for a 'New MacBook Pro' with a target of 2500CHF, showing a progress bar at 1500 of 2500 reached and a schedule of 'Put away 150CHF for this goal every month'. The second is for a 'Bike' with a target of 600CHF, showing a progress bar at 100 of 600 reached and a schedule of 'Put away 50CHF for this goal every month'. Below these is a '+ add new goal' button. The 'Overview' section on the right states: 'You'll reach all your saving goals (3100 CHF) within the next 12 months. You'll put 200 CHF every month towards your goals.'

# Make common tasks dead simple

When transferring money, one often has to enter the account details of the receiver again and again, why?  
 When a customer wants to check whether he already paid for something he has to scan his transactions, why?

- Make paying money dead simple: enter amount, recipient and let the system figure out the rest. Save the account details automatically in a list.
- Create task-oriented views for the transactions, for example let the customer view all transactions with a certain company or person.

The top screenshot shows the UBS 'Pay' interface. A red dashed box highlights the 'Pay' form where the amount '100' and currency 'CHF' are entered. A dropdown menu for recipients is open, showing 'Tho' (selected), 'Thomas Newman', 'Thompson Company', 'Thurstone', and '+ add new'. A red arrow points to this dropdown with the text 'paying money without having to remember account details'. Below the form are sections for 'Scheduled payments (3)' and 'Open eBills (6)'. A button 'Pay all checked bills (2819 CHF)' is visible.

The bottom screenshot shows the 'electro corp' account overview. A red dashed box highlights a table of 'payments to electro corp' with columns for Date, amount, and note. A red arrow points to this table with the text 'show all transactions with one company or person'. To the right, the 'recurring payments' section shows '350 CHF every two months' with a 'change' link and '+ add recurring payment schedule' link. A 'Make payment Request payment' button is also present.

paying money without having to remember account details

show recurring payment plans

show all transactions with one company or person

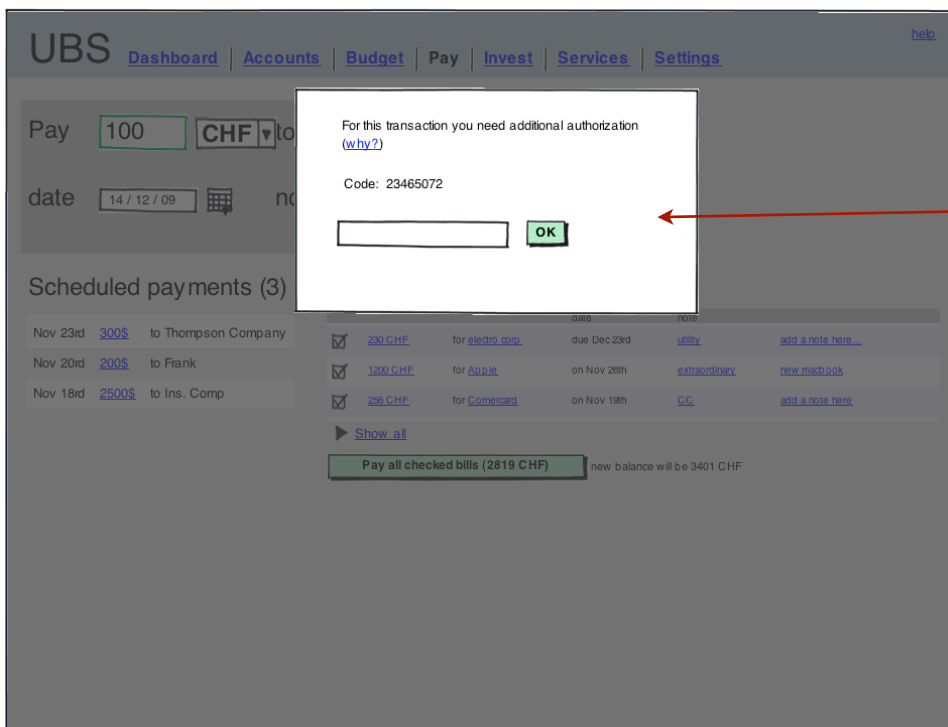
also allow the customer to pay to this account when the customers sees he didn't transfer the funds

# Easier access

Signing in with an access card and a card reader is a burden. This does not have to be this way.

Let the user do things like checking accounts, reviewing transactions, checking the budget without having him go through the access card / card reader process. As long as the user doesn't transfer funds, this layer of security is not needed. So only ask the user to use the card reader, when he actually wants to make a transaction or change a payment schedule. This way, using only login / password, the user can do a lot without having to go through the time-consuming login process.

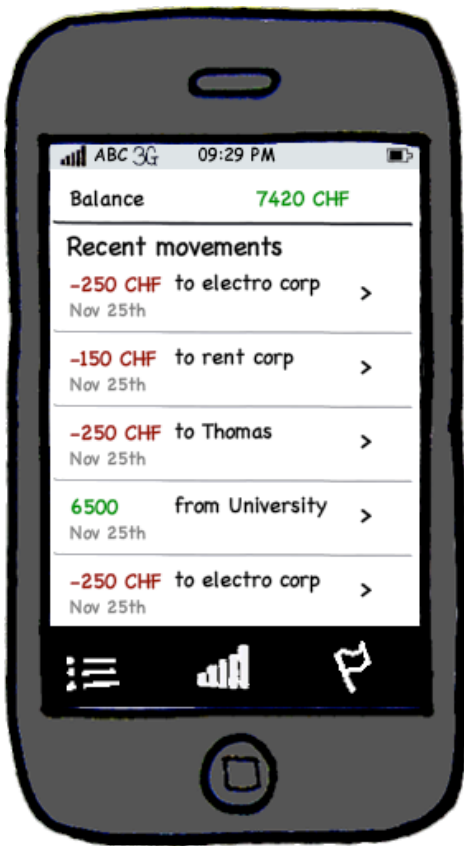
access level	authorization	access to
level 1 access	with login & password	checking balances, setting budget, allocating transfers to budget accounts, reviewing transactions
level 2 access	additional authorization with access card & card reader	all tasks involving transferring money



only once a customer tries to transfer funds, he is asked to use his card reader

## Enable control on the road

Tasks which don't require a high level of security could also be executed on mobile phones, this would be especially useful for the budgeting part of the application.



reviewing what's happening  
on your account



review your budget

# Development & implementation approach

Two approaches are possible:

One could create a web application in addition to the existing e-banking application, which would run in parallel. This would make sense in case the application would heavily emphasize features like budgeting. The very new structure might alienate some existing customers.

Or one could slowly evolve the existing application, introduce new views (for accounts or for all transactions with one company), and introduce features like saving goals and easy payments, dashboard functionality and so on.

Whatever is chosen the development & implementation process would be similar.

## 1. Identify the important goals and tasks for your different customer segments

This can be done using the methods for [mental models described by Indy Young](#). This means qualitative interviews with all the customer segments, then analyzing and grouping the mentioned goals and tasks, then aligning the features with those goals and tasks.

## 2. Build an interactive prototype

Using the usual technologies (HTML / CSS / Javascript)

## 3. Evolve the prototype using agile techniques and heavy user testing

Let us watch the customer using the prototype, then make changes according to the observations and let the customer try it again and again. No one knows better what works than your actual customers.

## 4. Extensive beta phase

Test it extensively in a beta phase with existing customers, find out what works and what doesn't work.

## 5. Deploy

## 6. Keep evolving

Ideas by

Roland Studer  
User experience Consultant  
avertas GmbH

**avertas**  
open & sustainable solutions